Tools for Large Graph Mining

Deepayan Chakrabarti

CMU-CALD-05-107

June 2005

Center for Automated Learning and Discovery
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Christos Faloutsos, Chair
Guy Blelloch,
Christopher Olston,
Jon Kleinberg, Cornell University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

This work is partially supported by the National Science Foundation under Grants No. CCR-0208853, ANI-0326472, IIS-0083148, IIS-0113089, IIS-0209107, IIS-0205224, INT-0318547, SENSOR-0329549, EF-0331657, IIS-0326322, CNS-0433540, IIS-9988876, by the Defense Advanced Projects Agency (DARPA) under USC subcontract no. IC-5003, by a generous fellowship from the Siebel Scholars Program, and by the Pennsylvania Infrastructure Technology Alliance, a partnership of Carnegie Mellon, Lehigh University, and the Commonwealth of Pennsylvania's Department of Community and Economic Development (DCED). Additional funding was provided by donations from Intel, and by a gift from Northrop-Grumman Corporation.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2005		**00-06-2005 to 00-06-2005**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Tools for Large Graph Mining	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,School of Computer Science,Pittsburgh,PA,15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**11**	

Keywords: Graphs, Viral propagation, Epidemic, Information survival threshold, Node grouping, Outlier detection, Graph generation

This thesis is dedicated to my parents M. K. Chakrabarti and K. Chakrabarti, without whose love and support, none of this would have been possible.

Abstract

Graphs show up in a surprisingly diverse set of disciplines, ranging from computer networks to sociology, biology, ecology and many more. How do such "normal" graphs look like? How can we spot abnormal subgraphs within them? Which nodes/edges are "suspicious?" How does a virus spread over a graph? Answering these questions is vital for outlier detection (such as terrorist cells, money laundering rings), forecasting, simulations (how well will a new protocol work on a realistic computer network?), immunization campaigns and many other applications.

We attempt to answer these questions in two parts. First, we answer questions targeted at *applications*: what patterns/properties of a graph are important for solving specific problems? Here, we investigate the propagation behavior of a computer virus over a network, and find a simple formula for the *epidemic threshold* (beyond which any viral outbreak might become an epidemic). We find an "information survival threshold" which determines whether, in a sensor or P2P network with failing nodes and links, a piece of information will survive or not. We also develop a scalable, parameter-free method for finding *groups* of "similar" nodes in a graph, corresponding to homogeneous regions (or *CrossAssociations*) in the binary adjacency matrix of the graph. This can help navigate the structure of the graph, and find un-obvious patterns.

In the second part of our work, we investigate recurring patterns in real-world graphs, to gain a deeper understanding of their structure. This leads to the development of the R-MAT model of graph generation for creating synthetic but "realistic" graphs, which match many of the patterns found in real-world graphs, including power-law and lognormal degree distributions, small diameter and "community" effects.

Acknowledgements

I would like to thank my advisor, Dr. Christos Faloutsos, for his constant support and encouragement. He has always found time to listen, to discuss, and to gently guide me onwards. He is not only an advisor, but a friend as well.

In addition, I would offer my thanks to Diane Stidle, who has been the heartbeat of CALD, and has smoothed the way for us all from start to finish; to fellow "CALDlings" including Ricardo, Edo, Leonid, Anya, Jason, and all others for their help and insights; and finally to Krishna, old friend, for his constant "Are you done yet?" which finally pushed me into getting "it" done.

Contents

List of Figures

List of Tables

Chapter 1

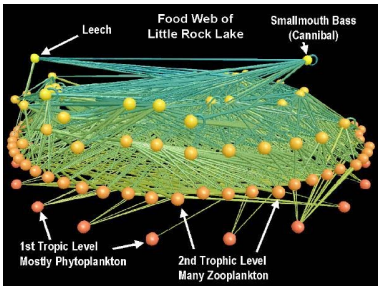
Introduction

Informally, a graph is a set of nodes, and a set of edges connecting some node pairs. In database terminology, the nodes represent individual entities, while the edges represent relationships between these entities. This formulation is very general and intuitive, which accounts for the wide variety of real-world datasets which can be easily expressed as graphs. Some examples include:

- *Computer Networks:* The Internet topology (at both the Router and the Autonomous System (AS) levels) is a graph, with edges connecting pairs of routers/AS. This is a self-graph, which can be both weighted or unweighted.
- *Ecology:* Food webs are self-graphs with each node representing a species, and the species at one endpoint of an edge eats the species at the other endpoint.
- *Biology:* Protein interaction networks link two proteins if both are necessary for some biological process to occur.
- *Sociology:* Individuals are the nodes in a social network representing ties (with *labels* such as friendship, business relationship, trust, etc.) between people.
- *User Psychology:* Clickstream graphs are bipartite graphs connecting Internet users to the websites they visit, thus encoding some information about the psyche of the web user.
- *Information Retrieval:* We can have bipartite graphs connecting "document" nodes to the "word" nodes that are present in that document. This link could be *weighted* by the number of occurrences of the word in the document. We also have *co-authorship* self-graphs, connecting authors who were co-authors of some document.

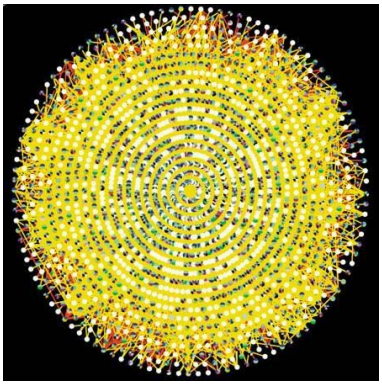
In fact, any information relating different entities (an $M : N$ relationship in database terminology) can be thought of as a graph. This accounts for the abundance of graphs in so many diverse topics of interest, most of them large and sparse. Figure 1.1 shows some such graphs. In our work, we focus on unweighted, unlabeled graphs, of both the self- and bipartite- graph types.

Given the widespread presence of such "graph" datasets, one obvious question becomes very important:

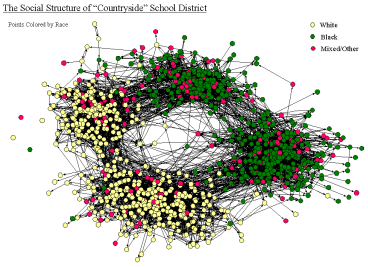


(a) Food web

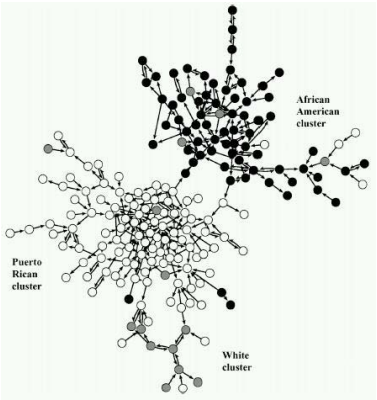
(b) Internet map



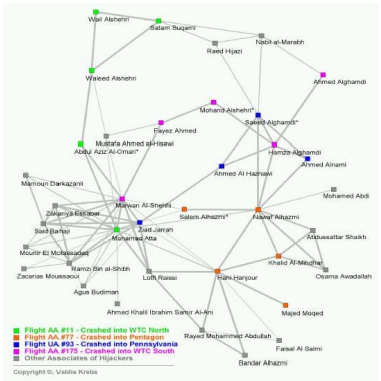
(c) Protein interactions



(d) Friendship network



(e) Needle-sharing network



(f) Hijackers network

Figure 1.1: *Examples of graphs:* (a) Food web [Martinez, 1991] (b) Internet topology map (`lumeta.com`) (c) Protein interaction network (`genomebiology.com`) (d) Friendship network among students in one American school [Moody, 2001] (e) "Needle-sharing" network of drug users [Weeks et al., 2002] (f) Network of the September-11 hijackers [Krebs, 2001].

How can we efficiently and accurately analyze large graph datasets, and mine them for interesting information?

Thanks to the generality of the graph representation of data, any new graph mining algorithm can be applied to a variety of datasets, perhaps obtained from completely different fields. Mining this information can provide significant benefits.

- *Security:* The "COPLINK Connect" system [Chen et al., 2003] has been used to combine and share data regarding criminals among many police departments. The criminal graph links suspects, crimes, locations, previous case histories, etc. These linkages provide the necessary information for effective law enforcement. Graph mining algorithms can also be used for finding abnormal subgraphs, say a money-laundering ring, in a large social network of financial transactions.

- *The World-wide Web:* To provide good results, a search engine must detect and counteract the "outliers" on the Web: spam sites, *"googlebombers"* [Google Bomb] and the like. Graph mining techniques are needed to automatically find such webpages out of the billions on the WWW.

- *Drug discovery:* Graph mining techniques can be used to search for *frequent subgraphs* in large chemical databases. For example, Dehaspe et al. [Dehaspe and Toivonen, 1999] look for patterns that regularly in carcinogenic compounds, and Milo et al. [Milo et al., 2002] detect "motifs" in genetic and other networks. Any information gleaned from such studies can be very useful in understanding diseases and developing drugs to counter them.

- *Immunizations:* Starting from one infected individual, a contagious disease can spread across a population (or computer network, for a computer virus) by infecting healthy individuals who are linked to infected ones. Thus, the structure of the "social network" plays a critical role in the spread of the disease, and in choosing the nodes whose immunization can "maximally disrupt" the contagion [Pastor-Satorras and Vespignani, 2002b].

- *Viral marketing:* This is exactly the opposite of immunizations: the aim is to target individuals who are best positioned in the social network to generate "word-of-mouth" publicity [Richardson and Domingos, 2002].

- *Biology:* Understanding the networks of regulatory genes and interacting proteins can give insights into the biological functions of these genes and proteins [Barabási, 2002].

Thus, there are many real-world applications where, given a specific graph, we want to find some properties or answer a particular query about it.

As against this, we can ask questions about real-world graphs in general, and not about any one graph in particular. We could ask:

What patterns occur regularly in real-world graphs? How do these graphs evolve over time? How fast do they grow?

The answers to these questions would be the "marks of realism" in real-world graphs, and we could use them to detect "fake" graphs or "abnormal" subgraphs. Knowledge of these patterns is also essential for another problem:

How can we build "realistic" yet efficient graph generators?

The criterion for judging the quality of a graph generator is the "realism" of the generated graphs: they are realistic exactly if they match the common graph patterns. Graph generators can in turn be used for a variety of purposes— graph sampling, graph compression, and graph extrapolation for simulation studies, to name a few.

Thus, we observe a dichotomy in graph mining applications: we can answer specific queries on a particular graph, or we can ask questions pertaining to real-world graphs in general. The separation between the two is, however, not strict, and there are applications requiring tools from both sides. For example, in order to answer "*how quickly will viruses spread on the Internet five years in the future*," we must have models for how the Internet will grow, how to generate a synthetic yet realistic graph of that size, and how to estimate the spread of viral infections on graphs.

In my thesis, I explore issues from both sides of this dichotomy. Since graphs are so general, these problems have been studied in several different communities, including computer science, physics, mathematics, physics and sociology. Often, this has led to independent rediscovery of the same concepts in different communities. In my work, I have attempted to combine these viewpoints and then improve upon them.

The specific problems I investigated in my research are as follows. Chapter 2 contains basic background material. Key symbols and terms used throughout the document are defined here. Any extra background material specific to each topic will be presented in the corresponding chapter. The next three chapters investigate applications of graph mining on *specific* graphs. In chapter 3, we analyze the problem of viral propagation in networks: *"Will a viral outbreak on a computer network spread to epidemic proportions, or will it quickly die out?"* We investigate the dependence of viral propagation on the network topology, and derive a simple and accurate *epidemic threshold* that determines if a viral outbreak will die out quickly, or survive for long in the network. In chapter 4, we study information survival in sensor networks: Consider a piece of information being spread within a sensor or P2P network with failing links and nodes. *What conditions on network properties determine if the information will survive in the network for long, or die out quickly?* In chapter 5, we answer the question: *How can we automatically find natural node groups in a large graph?* Our emphasis here is on a completely automatic and scalable system: the user only needs to feed in the graph dataset, and our Cross-associations algorithm determines both the number of clusters and their memberships. In addition, we present automatic methods for detecting outlier edges and for computing "inter-cluster distances."

Next, in chapter 6, we discuss issues regarding real-world graphs in general: *How can we quickly generate a synthetic yet realistic graph? How can we spot fake graphs and outliers?* We discuss several common graph patterns, and then present our R-MAT graph generator, which can match almost all these patterns using a very simple 3-parameter model. Finally, chapter 7 presents the conclusions of this thesis, followed by a discussion of future work in chapter 8.

Chapter 2

Basic Background Material

Before we discuss our graph mining tools, we must develop some basic terminology. We will start with the definition of a graph, and of the different types of graphs. We will then define the degree distribution of a graph, and briefly discuss "power laws." We will see the simple random graph model, which we will discuss in detail later in Chapter 6. Finally, we will define singular values and eigenvalues, which are extremely useful in the analysis of matrices and graphs, and will show up in later chapters.

2.1 Graph Types

The informal definition of a graph is a set of nodes with edges connecting some of them. However, there can be several variations on this theme, each of which has a special name. The graph can be directed (when edges point from one node to another) or undirected (edges point both ways). A graph can be a self-graph or a bipartite graph depending on whether the set of nodes pointed to by edges is the same as the set of nodes pointed from, or not. In addition, edges can have different weights, or not, which differentiates weighted graphs from unweighted graphs. All of these terms are formally defined below; figure 2.1 shows examples of these.

Definition 1 (Graph or Self-graph). *A* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *is a set* $\mathcal{V}$ *of* N *nodes, and a set* $\mathcal{E}$ *of* E *edges between them. The number of nodes is denoted by* $N = \|\mathcal{V}\|$*, and the number of edges by* $E = \|\mathcal{E}\|$.

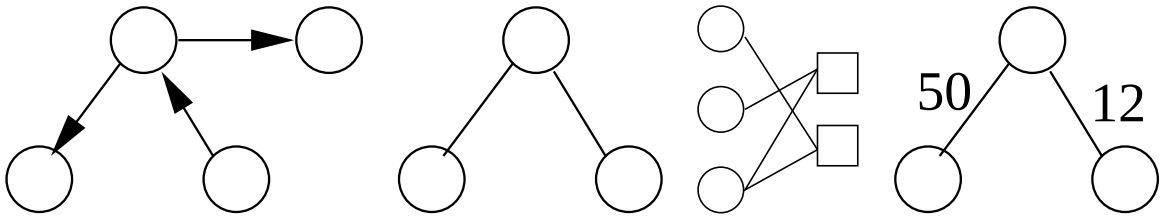
Definition 2 (Bipartite graph). *In a* bipartite graph*, the set of nodes* $\mathcal{V}$ *consists of two disjoint sets of nodes* $\mathcal{V}_1$ *and* $\mathcal{V}_2$*:* $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$*. Any edge connects a node in* $\mathcal{V}_1$ *to a node in* $\mathcal{V}_2$*, that is,* $(i, j) \in \mathcal{E} \Rightarrow i \in \mathcal{V}_1$ *and* $j \in \mathcal{V}_2$*. The number of nodes in the two node set are* $N_1 = \|\mathcal{V}_1\|$ *and* $N_2 = \|\mathcal{V}_2\|$*. The number of edges is still* $E = \|\mathcal{E}\|$.

Definition 3 (Directed and undirected graph). *In a directed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$*, each edge* $(i, j) \in \mathcal{E}$ *points from node* i *to node* j*. An undirected graph is a directed graph where edges point both ways, that is,* $(i, j) \in \mathcal{E} \Rightarrow (j, i) \in \mathcal{E}$.

Definition 4 (Weighted and unweighted graphs). *A weighted graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ *has a set of nodes* $\mathcal{V}$*, and a set of edges* $\mathcal{E}$*; and* $\mathcal{W}$ *represents the corresponding* weights *of those edges.*

Symbol	Description
G	The graph. $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for a weighted graph, and $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ for an unweighted graph
$\mathcal{V}$	The set of nodes in G. $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ for a bipartite graph
$\mathcal{E}$	The set of edges
$\mathcal{W}$	Edge weights: zero when no edge exists, and positive when it does. Edges weights are 1 for an unweighted graph.
$\mathbf{A}$	The adjacency matrix of the graph
N	Number of nodes in G
N_1, N_2	Number of nodes in the partitions $\mathcal{V}_1$ and $\mathcal{V}_2$ of a bipartite graph G
E	Number of edges in G

Table 2.1: *Table of basic symbols.*



(a) Directed self-graph (b) Undirected self-graph (c) Bipartite graph (d) Weighted self-graph

Figure 2.1: *Examples of different graph types:* (a) Edges in a directed graph point one way, such as a graph of papers where edges imply references from one paper to another. (b) Undirected graphs have edges pointing both ways, such as a social graph of kinship between individuals. Both of these represent self-graphs. (c) A bipartite graph has edges connecting two different sets of nodes, such as movies and the actors who acted in them. (d) A weighted graph has weights for each edge, such as the bandwidth of a link connecting two routers.

Weighted graphs can be both self-graphs or bipartite graphs. Unweighted graphs are a special case of weighted graphs, with all weights set to 1.

Definition 5 (Adjacency matrix of a self-graph). *The adjacency matrix* $\mathbf{A}$ *of a weighted self-graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ *is an* $N \times N$ *matrix such that*

$$\mathbf{A}_{i,j} = \begin{cases} w_{i,j} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \quad \text{for} \quad i,j \in 1 \ldots N$$

The adjacency for an unweighted self-graph merely replaces $w_{i,j}$ *by* 1.

Definition 6 (Adjacency matrix of a bipartite-graph). *The adjacency matrix* $\mathbf{A}$ *of a weighted bipartite-graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ *with* $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ *is an* $N_1 \times N_2$ *matrix such that*

$$\mathbf{A}_{i,j} = \begin{cases} w_{i,j} & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \quad \text{for} \quad \begin{matrix} i \in 1 \ldots N_1 \\ j \in 1 \ldots N_2 \end{matrix}$$

The adjacency for an unweighted bipartite-graph merely replaces $w_{i,j}$ *by* 1.

In addition, graph nodes and edges can have attached labels (i.e., categorical values); such graphs are called *labeled graphs*. However, our work has focused on unlabeled and unweighted graphs, both self- and bipartite, and both directed and undirected. Table 2.1 lists these symbols.

2.2 Degree Distributions and Power Laws

One basic property of nodes in the graph is their degree. We will define this, and the corresponding *degree distribution* of a graph. Many degree distributions in the real-world follow *power laws*, which we will touch upon next.

Definition 7 (Node Degree). *The outdegree of node* i *in graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *is the number of nodes it points to:*

$$\textit{Out-degree } d_{out}(i) \quad = \quad \|j \mid (i,j) \in \mathcal{E}\|$$

Similarly, the indegree is the number of nodes pointing to node i*:*

$$\textit{In-degree } d_{in}(i) \quad = \quad \|j \mid (j,i) \in \mathcal{E}\|$$

For an undirected graph, $d_{out}(i) = d_{in}(i)$ *for all nodes* i*.*

Definition 8 (Degree Distribution). *The degree distribution of an undirected graph is a plot of the count* c_k *of nodes with degree* k*, versus the degree* k*, typically on a log-log scale.*

Occasionally, the fraction $\frac{c_k}{N}$ is used instead of c_k; however, this merely translates the log-log plot downwards. For directed graphs, outdegree and indegree distributions are defined similarly.

Definition 9 (Power Law Degree Distributions). *A graph* $\mathcal{G}$ *has a power-law degree distribution if the number of nodes* c_k *with degree* k *is related to* k *by the equation:*

$$c_k = c \cdot k^{-\gamma} \tag{2.1}$$

where c *and* γ *are positive constants. In other words, the degree distribution looks linear when plotted on the log-log scale. The constant* γ *is often called the power law exponent.*

2.3 The Random Graph Model

The random graph model [Erdős and Rényi, 1960] is a famous and influential model of graph generation. We will discuss many other models in Chapter 6; however, this model will be needed earlier, and so we briefly mention it below.

Starting with a (user-provided) number of nodes N, the random graph generator [Erdős and Rényi, 1960] adds an edge between every pair of distinct nodes with a (user-provided) probability p. This simple model leads to a surprising list of properties, including phase transitions in the size of the largest component, and diameter logarithmic in graph size. Its ease of analysis has proven to be very useful in the early development of the field. Hence, many early graph-mining algorithms were focused on random graphs, and a basic idea of random graphs is useful when comparing our algorithms against previous methods.

2.4 Singular Value Decomposition

The technique of Singular Value Decomposition (SVD) is extremely useful in dealing with matrices. We will just present the basic concept here [Press et al., 1992]: "Any $M \times N$ matrix $\mathbf{A}$ whose number of rows M is greater than or equal to its number of columns N, can be written as the product of an $M \times N$ column-orthogonal matrix $\mathbf{U}$, an $N \times N$ diagonal matrix $\mathbf{W}$ with positive or zero elements (the *singular values*), and the transpose of an $N \times N$ orthogonal matrix $\mathbf{V}$."

$$\begin{pmatrix} & & \\ & \mathbf{A} & \\ & & \end{pmatrix} = \begin{pmatrix} & & \\ & \mathbf{U} & \\ & & \end{pmatrix} \begin{pmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_N \end{pmatrix} \begin{pmatrix} & & \\ & \mathbf{V^t} & \\ & & \end{pmatrix} \tag{2.2}$$

This decomposition can always be done; it provides an orthonormal *basis* for the rows and columns of the matrix $\mathbf{A}$; it is useful for obtaining least-squares fits to some matrix equations, for approximating matrices with just a few singular values, and for many other problems.

2.5 Eigenvalues

An $N \times N$ matrix $\mathbf{A}$ is said to have an *eigenvector* $\vec{x}$ and a corresponding *eigenvalue* λ if [Press et al., 1992]

$$\mathbf{A}\vec{x} = \lambda\vec{x} \tag{2.3}$$

These again provide an orthonormal basis for the matrix (indeed, eigenvalues and singular values are closely related), and they often show up in the analysis of matrices, as we shall see in later chapters.

Chapter 3

Epidemic thresholds in viral propagation

"Will a viral outbreak on a computer network spread to epidemic proportions, or will it quickly die out?"

The relevance of this question to the design of computer networks is obvious. Starting from some small corner of the Internet, modern viruses can quickly spread across the network. This network can be a physical, such as the network of routers, or it could be an "overlay" network, such the "addressbook" network (for viruses which spread via email). However, modern anti-virus techniques are typically "local": computers are disinfected one at a time, without considering the survival of the virus in the network. What is needed is method of combating viruses "globally," and that requires an understanding of the network-propagation behavior of viral outbreaks.

In addition to computer networks, the same questions appear in several other fields as well. Rumors, trends and the latest fads spread across a *social network*, and some fashion can gain prominence through "word of mouth" publicity. Dependable systems must be designed to prevent cascading failures. Immunization programs need to target individuals who are at a high risk, possibly due to their position in the social network. Similarly, the effectiveness of information dissemination or viral marketing campaigns depends on reaching "influential" individuals in the network. In all of these and many other cases, a good understanding of the propagation behavior over arbitrary network topologies is important.

Our contributions, as detailed in [Wang et al., 2003], are in answering the following questions:

- *How does a virus spread?* Specifically, we want an analytical model of viral propagation, that is applicable for *any* network topology.
- *When does the virus die out, and when does it become endemic?* Conceptually, a tightly connected graph offers more possibilities for the virus to spread and survive in the network than a sparser graph. Thus, the same virus might be expected to die out in one graph and become an epidemic in another. What features of the graph control this behavior? We find a simple closed-form expression for the "epidemic threshold" below which the virus dies out, but above which it can become an epidemic.
- *Below the threshold, how quickly will the virus die out?* A logarithmic decay of the virus

might still be too slow to have practical impact.

We will first present some background material relevant to this topic in Section 3.1. We then present our mathematical model of viral propagation in Section 3.2, and our derivation of the epidemic threshold condition in Section 3.3. Finally, we experimentally demonstrate the accuracy of our model in Section 3.4. Details of proofs are presented in Section 3.5, followed by a summary in Section 3.6.

3.1 Related Work

The process of viral propagation has been studied is several communities, including epidemiologists, computer scientists, physicists and probability theorists. The broad focus of all this work has been on two orthogonal aspects: the behavior of a single node in the graph, and the effect of network topology.

3.1.1 Behavior of a single node

How do individual nodes behave during a viral infection? The epidemiological community has studied this problem for a long time, and several models have been proposed [Bailey, 1975]. We will look at the most important ones below:

- *Susceptible-Infective-Susceptible (SIS) Model:* In this model, each node can be in one of two states: healthy but susceptible (S) to infection, or infected (I). An infected node can be cured locally (say, by anti-virus software), and it immediately becomes susceptible again. An infected node can also spread the virus along network links to neighboring susceptible nodes.

- *Susceptible-Infective-Removed (SIR) Model:* Here, once an infected node is cured, it is immune to further infections and is removed (R) from the network.

- *Extensions:* Several extensions to these basic models have been proposed. Two common ideas are "infection delay" and "user vigilance" [Wang and Wang, 2003]. Infection delay represents a delay in spreading the virus from an infected node. User vigilance is some time period after a disinfection when the user is vigilant against new infections, reducing the susceptibility of that node.

While each model is suitable for different applications, we focus on the plain SIS model, and all our future discussions will be based on it. The virus is modeled by two parameters: the virus "birth rate" β, with which it tries to spread across a network link to a susceptible node; and the virus "death rate" δ with which each infected node disinfects itself. If death rate is very low, the infection should survive for a long time, but if the birth rate is very low, it should die off quickly. Is there some condition/threshold that determines the fate of a viral outbreak?

This is called the "epidemic threshold" of the network, and has been the focus of a lot of research activity. Informally, the epidemic threshold is a value τ such that

$$\begin{aligned} &\text{If } \tfrac{\beta}{\delta} < \tau, && \text{the viral outbreak dies out quickly, but} \\ &\text{if } \tfrac{\beta}{\delta} \geq \tau, && \text{the virus may become endemic.} \end{aligned} \tag{3.1}$$

We shall provide a formal definition later in the text.

3.1.2 Interactions between nodes

While the previous paragraphs described the spread of a virus in the neighborhood of an infected node, the epidemic threshold is a global value depending on the topology of the entire network. This question has been approached from several viewpoints, of which the major ones we discuss below.

The KW model: Kephart and White [Kephart and White, 1991, 1993] modeled the network as directed graph, but with a constrained topology (either an Erdős-Rényi random graph, or a tree, or a 2D lattice). For this model (which we will call KW), they derived steady-state solutions for the number of infected nodes η_{KW}; for example, for the random graph, it is:

$$\eta_{KW} = N\left(1 - \frac{\delta}{\beta\langle k\rangle}\right) \tag{3.2}$$

where N is the total number of nodes, and $\langle k\rangle$ the average degree. They also derived the epidemic threshold:

$$\tau_{KW} = \frac{1}{\langle k\rangle} \tag{3.3}$$

While this is a good model for networks where contact among nodes is sufficiently homogeneous, there is overwhelming evidence that real networks (including social networks [Domingos and Richardson, 2001], router and AS networks [Faloutsos et al., 1999], and Gnutella overlay graphs [Ripeanu et al., 2002]) deviate from such homogeneity—they follow a power law structure instead. In such graphs, there exist a few nodes with very high connectivity, but the majority of the nodes have low connectivity. The high-connectivity nodes are expected to often get infected and then propagate the virus, making the infection harder to eradicate. We shall show that the KW model is a special case of our model, and we match its predictions for homogeneous graphs.

The MFA model (Mean Field Assumption model): Pastor-Satorras and Vespignani [Pastor-Satorras and Vespignani, 2001, Pastor-Satorras et al., 2001, Pastor-Satorras and Vespignani, 2002a,b] allowed the network topology to be a power-law. To derive analytic results, they used the mean-field assumption (MFA), where all nodes with the same degree are treated equally. We call this the "MFA" model. For the special case of the Barabási-Albert power-law topology [Barabási and Albert, 1999], their steady-state infected population size is:

$$\eta_{MFA} = 2Ne^{-\delta/m\beta}$$

where m is the minimum degree in the network. However, this has not been extended to all power-law graphs in general. They also derive an epidemic threshold:

$$\tau_{MFA} = \frac{\langle k\rangle}{\langle k^2\rangle} \tag{3.4}$$

where $\langle k \rangle$ is the average degree, and $\langle k^2 \rangle$ the average of the squared degree.

While the mean-field assumption lets us compute a general epidemic threshold, there is no reason for any two nodes with the same degree to be equally affected by the virus. One node could be in the core of a network, and another in the periphery; the one in the core will presumably be infected more often. Indeed, we observe experimentally that our model yields more accurate predictions than the MFA model.

Correlated networks: Boguñá and Satorras [Boguñá and Pastor-Satorras, 2002] studied networks where the connectivity of a node is related to the connectivity of its neighbors. These *correlated networks* include Markovian networks where, in addition to the degree distribution $P(k)$, a function $P(k|k')$ determines the probability that a node of degree k is connected to a node of degree k'.

While some degree of correlations may exist in real networks, it is often difficult to characterize connectivity correlations with a simple $P(k|k')$ function. Computing these values in real-world graphs with strong confidence guarantees is also hard. In addition, our results indicate that knowledge of $P(k|k')$ is not needed to compute the epidemic threshold for arbitrary graphs.

Interacting particle systems: This is a branch of probability theory where "particles" are allowed to propagate over a simple network according to different processes; the one that is closest to our work is the "contact process" [Harris, 1974, Liggett, 1985]. This area has seen some excellent and rigorous theoretical work. However, the networks that are studied are (a) infinite, which changes the questions being asked, and (b) simple, typically only line graphs and grids [Durrett and Liu, 1988, Durrett and Schonmann, 1988, Durrett et al., 1989]. Instead, the (possibly arbitrary) network topology is a central part of our work. Asavathiratham [Asavathiratham, 2000] studies similar processes, but does not investigate epidemic thresholds.

Thus, all current methods either constrain the network topology or use strong assumptions. Next, we propose a general method of modeling viral propagation on *any finite graph*, without having to resort to the mean-field approximation.

3.2 Model of Viral Propagation

We develop a mathematical model for the SIS method of viral infection, which is applicable to any undirected graph G. Our model assumes very small discrete timesteps of size Δt, where $\Delta t \to 0$. Our results apply equally well to continuous systems, though we focus on discrete systems for ease of exposition. Within a Δt time interval, an infected node i tries to infect its neighbors with probability β. At the same time, i may be cured with probability δ. Table 3.1 defines these and other symbols.

3.2.1 Viral propagation as a Markov chain

The spread of the virus is governed by a Markov chain with 2^N states. Each state in the Markov chain corresponds to one particular system configuration of N nodes, each of which can be in

one of two states (Susceptible or Infective), which leads to 2^N possible configurations. Also, the configuration at time-step $t+1$ depends only on that at time-step t; thus, it is a Markov chain.

This Markov chain also has an "absorbing" state, when all nodes are uninfected (i.e., Susceptible). This absorbing state can be reached from any starting state of the Markov chain, implying that this state will be reached with probability 1 over a long period of time. However, this state could be reached very quickly, or it could take time equivalent to the age of the universe (in which case, the viral epidemic practically never dies).

3.2.2 Main Idea

The obvious approach of solving the Markov chain becomes infeasible for large N, due to the exponential growth in the size of the chain. To get around this limitation, we use the "independence" assumption, that is, the states of the neighbors of any given node are independent. Thus, we replace the problem with Equation 3.6 (our "non-linear dynamical system" discussed below), with only N variables instead of 2^N for the full Markov chain. This makes the problem tractable, and we can find closed-form solutions.

Note that the independence assumption places no constraints on network topology; our method works with any *arbitrary* finite graph. Also, this assumption is far less constraining than the mean-field assumption. In fact, as we will show experimentally, the number of infected nodes over time under the "independence assumption" is very close to the that without any assumptions, for a wide range of datasets.

3.2.3 Mathematical formulation

Let the probability that a node i is infected at time t by $p_i(t)$. Let $\zeta_i(t)$ be the probability that a node i will not receive infections from its neighbors in the next time-step. This happens if each neighbor is either uninfected, or is infected but fails to spread the virus with probability $(1-\beta)$:

$$\begin{aligned}\zeta_i(t) &= \prod_{j:neighbor\ of\ i} (p_j(t-1)(1-\beta) + (1-p_j(t-1))) \\ &= \prod_{j:neighbor\ of\ i} (1-\beta * p_j(t-1)) \end{aligned} \tag{3.5}$$

This is the *independence assumption:* we assume that probabilities $p_j(t-1)$ are independent of each other.

A node i is healthy at time t if it did not receive infections from its neighbors at t *and* i was uninfected at time-step $t-1$, or was infected but was cured at t. Denoting the probability of a node i being infected at time t by $p_i(t)$:

$$1-p_i(t) = (1-p_i(t-1)) \cdot \zeta_i(t) + \delta \cdot p_i(t-1) \cdot \zeta_i(t) \quad i = 1 \ldots N \tag{3.6}$$

This equation represents our *NLDS* (Non-Linear Dynamical System). Figure 3.1 shows the transition diagram.

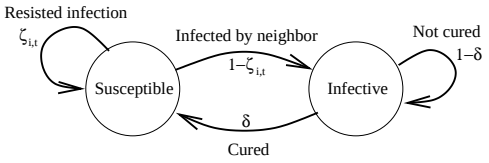


Figure 3.1: *The SIS model, as seen from a single node:* Each node, in each time step t, is either Susceptible (S) or Infective (I). A susceptible node i is currently healthy, but can be infected (with probability $1 - \zeta_{i,t}$) on receiving the virus from a neighbor. An infective node can be cured with probability δ; it then goes back to being susceptible. Note that $\zeta_{i,t}$ depends on the both the virus birth rate β and the network topology around node i.

Symbol	Description
$\mathcal{G}$	An undirected connected graph
N	Number of nodes in $\mathcal{G}$
E	Number of edges in $\mathcal{G}$
$\mathbf{A}$	Adjacency matrix of $\mathcal{G}$
$\mathbf{A}'$	The transpose of matrix $\mathbf{A}$
β	Virus birth rate on a link connected to an infected node
δ	Virus death rate on an infected node
t	Time stamp
$p_i(t)$	Probability that node i is infected at t
$\vec{p}(t)$	$\vec{p}(t) = (p_1(t), p_2(t), \ldots, p_N(t))'$
$\zeta_i(t)$	Probability that node i does not receive infections from its neighbors at t
$\lambda_{i,A}$	The i-th largest eigenvalue of $\mathbf{A}$
$\vec{u}_{i,A}$	Eigenvector of $\mathbf{A}$ corresponding to $\lambda_{i,A}$
$\vec{u}_{i,A}{}'$	Transpose of $\vec{u}_{i,A}$
$\mathbf{S}$	The 'system' matrix describing the equations of infection
$\lambda_{i,S}$	The i-th largest eigenvalue of $\mathbf{S}$
η_t	Number of infected nodes at time t
$\langle k \rangle$	Average degree of nodes in a network
$\langle k^2 \rangle$	Connectivity divergence (average of squared degrees)

Table 3.1: Table of Symbols

We have thus turned the Markov chain into a non-linear dynamical system (Equation 3.6), and we will focus on its behavior. Specifically, we will find the epidemic threshold for this system. An informal definition of the epidemic threshold was presented in Equation 3.1; we will now present a formal definition for the *NLDS* case.

Definition 10 (Epidemic threshold under *NLDS*). *The epidemic threshold τ_{NLDS} for* NLDS *is a value such that*

$$\begin{aligned} \beta/\delta < \tau_{NLDS} \quad &\Rightarrow \quad \textit{infection dies out over time, } p_i(t) \to 0 \textit{ as } t \to \infty \ \forall i \\ \beta/\delta > \tau_{NLDS} \quad &\Rightarrow \quad \textit{infection survives and becomes an epidemic} \end{aligned}$$

3.3 The Epidemic Threshold

In this section, we shall discuss our simple closed-form formula for τ_{NLDS}. Surprisingly, it depends only on one number: the largest eigenvalue of the graph. Then, we will prove that below the threshold (i.e., when the viral outbreak dies), the decay is exponential. Finally, we will present some corollaries and special cases, which demonstrate the intuitiveness of the result.

Theorem 1 (Epidemic Threshold). *In* NLDS, *the epidemic threshold τ_{NLDS} for an undirected graph is*

$$\boxed{\tau_{NLDS} = \frac{1}{\lambda_{1,A}}} \tag{3.7}$$

where $\lambda_{1,A}$ is the largest eigenvalue of the adjacency matrix $\mathbf{A}$ of the network.

Proof. We will prove this in two parts: the necessity of this condition in eliminating an infection, and the sufficiency of this condition for wiping out *any* initial infection. The corresponding theorem statements are shown below; the proofs are described in Section 3.5. Following this, we will see how quickly an infection dies out if the epidemic threshold condition is satisfied. Recent follow-up work on this problem by Ganesh et al. [Ganesh et al., 2005] has confirmed our results, and added more results about system behavior *above* the epidemic threshold. □

Theorem 2 (Part A: Necessity of Epidemic Threshold). *In order to ensure that over time, the infection probability of each node in the graph goes to zero (that is, the epidemic dies out), we must have $\frac{\beta}{\delta} < \tau_{NLDS} = \frac{1}{\lambda_{1,A}}$.*

Proof. Proved in Section 3.5. □

Theorem 3 (Part B: Sufficiency of Epidemic Threshold). *If $\frac{\beta}{\delta} < \tau_{NLDS} = \frac{1}{\lambda_{1,A}}$, then the epidemic will die out over time (the infection probabilities will go to zero), irrespective of the size of the initial outbreak of infection.*

Proof. Proved in Section 3.5. □

Definition 11 (Score). *Score $s = \frac{\beta}{\delta} \cdot \lambda_{1,A}$.*

Theorem 1 provides the conditions under which an infection dies out ($s < 1$) or survives ($s \geq 1$) in our dynamical system. To visualize this, consider the spread of infection as a random walk on the graph. The virus spreads across one hop according to $\beta\mathbf{A}$, and thus it spreads across h hops according to $(\beta\mathbf{A})^h$, which grows as $(\beta\lambda_{1,A})$ every hop. On the other hand, the virus dies off at a rate δ. Thus, the "effective" rate of spread is approximately $\beta\lambda_{1,A}/\delta$, which is exactly the "score" s. Thus, to have any possibility of an epidemic, the score s must be greater than 1. This is exactly the epidemic threshold condition that we find.

We can ask another question: if the system is below the epidemic threshold, how *quickly* will an infection die out?

Theorem 4 (Exponential Decay). *When an epidemic is diminishing* (*therefore* $\beta/\delta < \frac{1}{\lambda_{1,A}}$ *and* $s < 1$), *the probability of infection decays at least exponentially over time.*

Proof. Proved in Section 3.5. □

We can use Theorem 1 to compute epidemic thresholds for many special cases, as detailed below. All of these are proved in Section 3.5.

Corollary 1. NLDS *subsumes the KW model for homogeneous or random Erdős-Rényi graphs.*

Corollary 2. *The epidemic threshold* τ_{NLDS} *for a star topology, is exactly* $\frac{1}{\sqrt{d}}$, *where* $\sqrt{d}$ *is the square root of the degree of the central node.*

Corollary 3. *The epidemic threshold for an infinite power-law network is zero.*

Corollary 4. *Below the epidemic threshold (score* $s < 1$*), the expected number of infected nodes* η_t *at time* t *decays exponentially over time.*

"Optimistic" versus "Pessimistic" scenarios: The SIS model is a "pessimistic" model of viral infection: no node is ever immunized, and a cured node is immediately susceptible, making it very hard for us to stop the viral spread. As against this, we can think of the SIR model as the "optimistic" model. Thus, if we design a network topology to be resistant to viral epidemics at some epidemic threshold (given by Theorem 1) for the SIS model, it should be resistant even under the SIR model.

Again, irrespective of the model, we can have different starting conditions for the viral infection: one node infected at time $t = 0$, all nodes infected, or somewhere in between. Theorem 1 holds even in the pessimistic case where all nodes start off infected: if we are below the threshold (score $s < 1$), then the epidemic dies out quickly regardless of the starting state.

Thus, since we work under a pessimistic scenario, our results will hold for many other models and conditions. The price for this generality is that our results may be too conservative; perhaps less stringent conditions would suffice to drive the virus extinct in optimistic scenarios. There may also be other applications where the optimistic scenario makes more sense. These are all interesting questions, and we intend to study them in the future.

Dataset	Nodes	Edges	Largest Eigenvalue
RANDOM	256	982	8.691
POWER-LAW	3,000	5,980	11.543
STAR-10K	10,001	10,000	100
OREGON	11,461	32,730	75.241

Table 3.2: Dataset characteristics.

3.4 Experiments

We want to answer the following questions:

(Q1) How closely does our dynamical system model the spread of a viral infection?

(Q2) How accurate is our epidemic threshold condition?

(Q3) How does our epidemic threshold compare to those suggested in [Pastor-Satorras and Vespignani, 2002a]?

(Q4) Below the threshold, does the epidemic die out exponentially quickly?

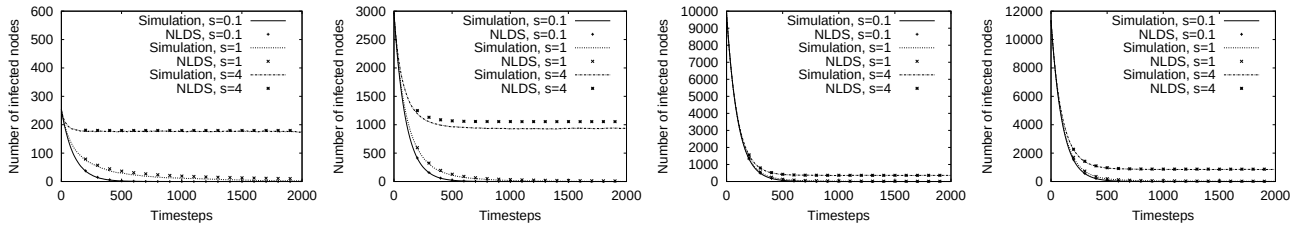
The datasets we used were:

- *RANDOM*: An Erdős-Rényi random graph of 256 nodes and 982 edges.
- *POWER-LAW*: A graph of $3,000$ nodes and $5,980$ edges, generated by the popular Barabási-Albert process [Barabási and Albert, 1999]. This generates a graph with a power-law degree distribution of exponent 3.
- *STAR-10K*: A "star" graph with one central hub connected to $10,000$ "satellites."
- *OREGON*: A real-world graph of network connections between Autonomous Systems (AS), obtained from `http://topology.eecs.umich.edu/data.html`. It has $11,461$ nodes and $32,730$ edges.

For each dataset, all nodes were initially infected with the virus, and then its propagation was studied in a simulator. All simulations were run for $10,000$ timesteps, and were repeated 100 times with different seeds. Table 3.2 provides more details.

3.4.1 (Q1) Accuracy of dynamical system

In figure 3.2, we plot the number of infected nodes over time, and compare it against the evolution of *NLDS* (Equation 3.6). Several values of the score s were used (below, at and above the threshold). In all cases, the simulation results are extremely close to those from *NLDS*. Thus, *NLDS* is a good approximation of the true system.



(a) *RANDOM* (b) *POWER-LAW* (c) *STAR-10K* (d) *OREGON*

Figure 3.2: *Simulation versus dynamical system:* The number of infected nodes is plotted versus time, for both simulations (lines) and *NLDS* (points). Observe the close fit between the two.

Thus, *the dynamical system of equation 3.6 is a very good representation of viral propagation in arbitrary networks.*

3.4.2 (Q2) Accuracy of epidemic threshold

Figure 3.3 shows the number of infected nodes over time for various values of the score s, in log-log scales. We observe a clear trend: below the threshold ($s < 1$), the infection dies off, while it survives above the threshold ($s > 1$). This is exactly as predicted by Theorem 1, and justifies our formula for the threshold.

Thus, *our epidemic threshold condition is accurate: infections become extinct below the threshold, and survive above it.*

3.4.3 (Q3) Comparison with previous work

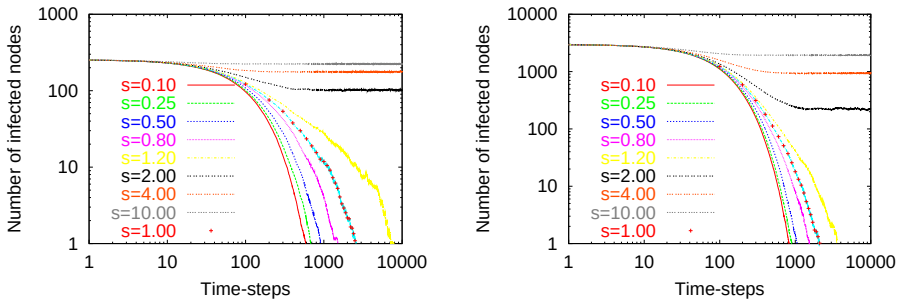
In figure 3.4, we compare the predicted threshold of our model against that of the MFA model. For several values of the score s, we plot the number of infected nodes left after a "long" time (specifically $500, 1000$ and 2000 timesteps). Below the threshold, the infection should have died out, while it could have survived above the threshold. In all cases, we observe that this change in behavior (extinction to epidemic) occurs at our predicted epidemic threshold.

Note that the *NLDS* threshold is more accurate than that of the MFA model for the *STAR-10K* and the real-world *OREGON* graphs, while we subsume their predictions for *RANDOM* and *POWER-LAW*, which are the topologies the MFA model was primarily developed for.

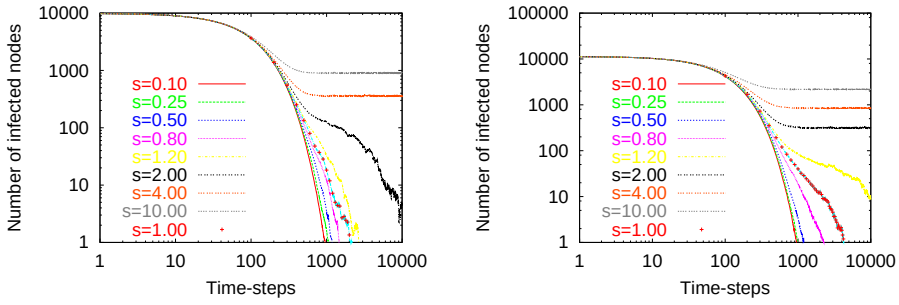
Recalling also that our model subsumes the KW model on homogeneous networks (Corollary 1), we arrive at the following conclusion: *our epidemic threshold for* NLDS *subsumes or performs better than those for other models.*

3.4.4 (Q4) Exponential decay of infection under threshold

Figure 3.5 demonstrates the rate of decay of the infection when the system is below the threshold ($s < 1$). The number of infected nodes is plotted against time on a log-linear scale We see that

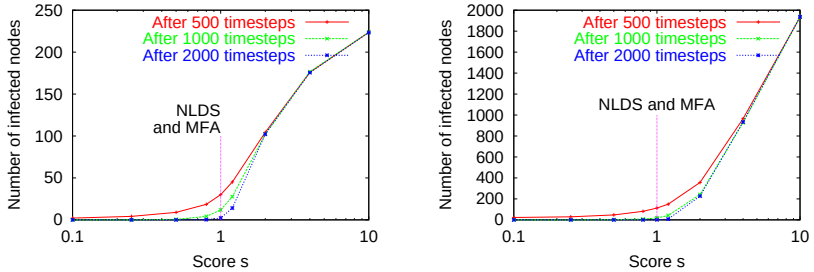


(a) *RANDOM* (b) *POWER-LAW*

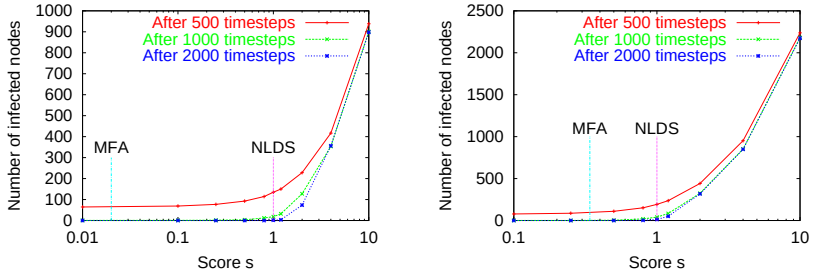


(c) *STAR-10K* (d) *OREGON*

Figure 3.3: *Accuracy of our epidemic threshold:* The number of infected nodes is plotted versus time for various values of the score s (log-log scales). The case when score $s = 1$ is shown with the dotted line. There is a clear distinction between the cases where $s < 1$ and $s > 1$: below 1, the infection dies out quickly, while above 1, it survives in the graph. This is exactly our proposed epidemic threshold condition.

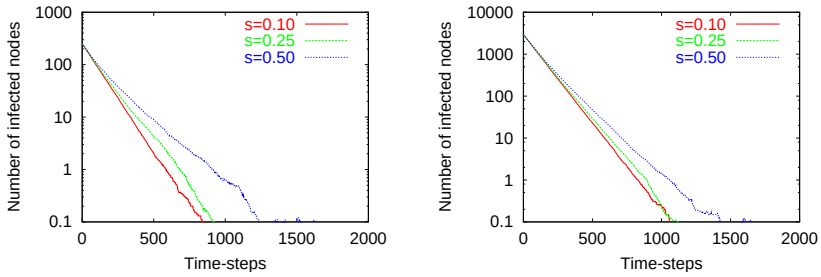


(a) *RANDOM* (b) *POWER-LAW*



(c) *STAR-10K* (d) *OREGON*

Figure 3.4: *Comparison with the MFA model:* We plot number of infected nodes after a "long" time for various values of the score s, versus s. For each dataset, we show results after $500, 1000$ and 2000 timesteps. In each case, we observe a sharp jump in the size of the infected population at our epidemic threshold of $s = 1$. Note that our results are far more accurate than those of the MFA model.



(a) *RANDOM* (b) *POWER-LAW*

(c) *STAR-10K* (d) *OREGON*

Figure 3.5: *Exponential decay below the threshold:* The number of infected nodes is plotted versus time for various values of the score $s < 1$ (log-linear scales). Clearly, the decay is exponentially quick (because it is linear on a log-linear plot). This matches the predictions of Theorem 4.

in all cases, the decay is exponential (because the plot looks linear on a log-linear scale). This is exactly as predicted by Theorem 4.

Thus, *the infection dies out exponentially quickly below the threshold ($s < 1$).*

3.5 Details of Proofs

Theorem 1 (Epidemic Threshold). *In* NLDS*, the epidemic threshold τ_{NLDS} for an undirected graph is*

$$\boxed{\tau_{NLDS} = \frac{1}{\lambda_{1,A}}} \tag{3.8}$$

where $\lambda_{1,A}$ is the largest eigenvalue of the adjacency matrix $\mathbf{A}$ of the network.

Proof. The proof follows from the proofs of Theorems 2 and 3 below. □

Theorem 2 (Part A: Necessity of Epidemic Threshold). *In order to ensure that over time, the infection probability of each node in the graph goes to zero (that is, the epidemic dies out), we must have $\frac{\beta}{\delta} < \tau_{NLDS} = \frac{1}{\lambda_{1,A}}$.*

Proof. We have modeled viral propagation as a *discrete dynamical system*, with the following non-linear dynamical equation:

$$1 - p_i(t) \quad = \quad (1 - p_i(t-1)) \cdot \zeta_i(t) + \delta \cdot p_i(t-1) \cdot \zeta_i(t) \qquad \text{(from Eq 3.6)}$$

$$\begin{aligned} \text{or, } \vec{p}(t) &= g\left(\vec{p}(t-1)\right) \\ \text{where, } \quad g_i\left(\vec{p}(t-1)\right) &= 1-(1-p_i(t-1))\cdot\zeta_i(t)-\delta\cdot p_i(t-1)\cdot\zeta_i(t) \end{aligned} \tag{3.9}$$

where $g:[0,1]^N \rightarrow [0,1]^N$ is a function on the probability vector, and $g_i(.)$ is its i-th element.

The infection dies out when $p_i = 0$ for all i. We can easily check that the $\vec{p} = \vec{0}$ vector is a *fixed point* of the system; when $p_i(t-1) = 0$ for all i (all nodes healthy), the equation above results in $p_i(t) = 0$ for all i, and so all nodes remain healthy forever.

The question we need to answer is: If the infection probabilities of all nodes in the graph come close to zero, will the dynamical system push them even closer to zero? If yes, the infection probabilities will go to zero and the infection will die out, but if not, the infection could survive and become an epidemic. This concept is known as *asymptotic stability*, and conditions for achieving asymptotic stability are well-known.

Definition 12 (Asymptotic Stability of a Fixed Point). *A fixed point P_f is "asymptotically stable" if, on a slight perturbation from P_f, the system returns to P_f (as against moving away, or staying in the neighborhood of P_f but not approaching it) [Hirsch and Smale, 1974].*

Lemma 1 (Condition for Asymptotic stability). *The system is asymptotically stable at $\vec{p} = \vec{0}$ if the eigenvalues of $\nabla g(\vec{0})$ are less than 1 in absolute value. Recall that $\nabla g\left(\vec{x}\right) = \left[\frac{\partial g_i}{\partial x_j}\right]$ for $i,j = 1\ldots N$, and thus, $\left[\nabla g(\vec{0})\right]_{i,j} = \frac{\partial g_i}{\partial p_j}\Big|_{\vec{p}=\vec{0}}$. Proved in [Hirsch and Smale, 1974].*

From Eq 3.9, we can calculate $\nabla g(\vec{0})$:

$$\begin{aligned} \left[\nabla g(\vec{0})\right]_{i,j} &= \begin{cases} \beta\mathbf{A}_{j,i} & \text{for} \quad j \neq i \\ 1-\delta & \text{for} \quad j = i \end{cases} \\ \text{Thus,} \quad \nabla g(\vec{0}) &= \beta\mathbf{A}' + (1-\delta)\mathbf{I} \\ &= \beta\mathbf{A} + (1-\delta)\mathbf{I} \end{aligned} \tag{3.10}$$

where the last step follows because $\mathbf{A} = \mathbf{A}'$ (since the graph is undirected).

This matrix describes the behavior of the virus when it is very close to dying out; we call it the system matrix $\mathbf{S}$:

$$\mathbf{S} = \nabla g(\vec{0}) = \beta\mathbf{A} + (1-\delta)\mathbf{I} \tag{3.11}$$

As shown in Lemma 2 following this proof, the matrices $\mathbf{A}$ and $\mathbf{S}$ have the same eigenvectors $\vec{u}_{i,S}$, and their eigenvalues, $\lambda_{i,A}$ and $\lambda_{i,S}$, are closely related:

$$\lambda_{i,S} = 1-\delta+\beta\lambda_{i,A} \quad \forall i \tag{3.12}$$

Hence, using the stability condition above, the system is asymptotically stable when

$$|\lambda_{i,S}| < 1 \quad \forall i \tag{3.13}$$

that is, all eigenvalues of $\mathbf{S}$ have absolute value less than one.

Now, since $\mathbf{A}$ is a real symmetric matrix (because the graph is undirected), its eigenvalues $\lambda_{i,A}$ are real. Thus, the eigenvalues of $\mathbf{S}$ are real too. Also, since the graph $\mathcal{G}$ is a connected undirected

graph, the matrix $\mathbf{A}$ is a real, nonnegative, irreducible, square matrix. Under these conditions, the Perron-Frobenius Theorem [MacCluer, 2000] says that the largest eigenvalue is a positive real number and also has the largest magnitude among all eigenvalues. Thus,

$$\lambda_{1,S} = |\lambda_{1,S}| \geq |\lambda_{i,S}| \quad \forall i \tag{3.14}$$

Using this in Equation 3.13:

$$\begin{aligned} & \lambda_{1,S} < 1 \\ \text{that is,} \quad & 1 - \delta + \beta\lambda_{1,A} < 1 \end{aligned}$$

which means that an epidemic is prevented if $\beta/\delta < 1/\lambda_{1,A}$. Also, if $\beta/\delta > 1/\lambda_{1,A}$, the probabilities of infection may diverge from zero, and an epidemic could occur. Thus, the epidemic threshold is $\boxed{\tau_{NLDS} = \frac{1}{\lambda_{1,A}}}$

□

Lemma 2 (Eigenvalues of the system matrix). *The $i-th$ eigenvalue of $\mathbf{S}$ is of the form $\lambda_{i,S} = 1 - \delta + \beta\lambda_{i,A}$, and the eigenvectors of $\mathbf{S}$ are the same as those of $\mathbf{A}$.*

Proof. Let $\mathbf{u_{i,A}}$ be the eigenvector of $\mathbf{A}$ corresponding to eigenvalue $\lambda_{i,A}$. Then, by definition, $\mathbf{A}\vec{\mathbf{u}}_{\mathbf{i,A}} = \lambda_{i,A} \cdot \vec{\mathbf{u}}_{\mathbf{i,A}}$ Now,

$$\begin{aligned} \mathbf{S}\vec{\mathbf{u}}_{\mathbf{i,A}} &= (1-\delta)\mathbf{u}_{\mathbf{i,A}} + \beta\mathbf{A}\vec{\mathbf{u}}_{\mathbf{i,A}} \\ &= (1-\delta)\vec{\mathbf{u}}_{\mathbf{i,A}} + \beta\lambda_{i,A}\vec{\mathbf{u}}_{\mathbf{i,A}} \\ &= (1-\delta+\beta\lambda_{i,A})\vec{\mathbf{u}}_{\mathbf{i,A}} \end{aligned} \tag{3.15}$$

Thus, $\vec{\mathbf{u}}_{\mathbf{i,A}}$ is also an eigenvector of $\mathbf{S}$, and the corresponding eigenvalue is $(1 - \delta + \beta\lambda_{i,A})$.

Conversely, suppose $\lambda_{i,S}$ is an eigenvalue of $\mathbf{S}$ and $\mathbf{u_{i,S}}$ is the corresponding eigenvector. Then,

$$\begin{aligned} \lambda_{i,S}\vec{\mathbf{u}}_{\mathbf{i,S}} &= \mathbf{S}\vec{\mathbf{u}}_{\mathbf{i,S}} \\ &= (1-\delta)\vec{\mathbf{u}}_{\mathbf{i,S}} + \beta\mathbf{A}\vec{\mathbf{u}}_{\mathbf{i,S}} \\ \Rightarrow \left(\frac{\lambda_{i,S}+\delta-1}{\beta}\right)\vec{\mathbf{u}}_{\mathbf{i,S}} &= \mathbf{A}\vec{\mathbf{u}}_{\mathbf{i,S}} \end{aligned}$$

Thus, $\vec{u}_{i,S}$ is also an eigenvector of $\mathbf{A}$, and the corresponding eigenvalue of $\mathbf{A}$ is $\lambda_{i,A} = (\lambda_{i,S} + \delta - 1)/\beta$. □

Theorem 3 (Part B: Sufficiency of Epidemic Threshold). *If $\frac{\beta}{\delta} < \tau_{NLDS} = \frac{1}{\lambda_{1,A}}$, then the epidemic will die out over time (the infection probabilities will go to zero), irrespective of the size of the initial outbreak of infection.*

Proof.

$$\begin{aligned} \zeta_i(t) &= \prod_{j:neighbor\ of\ i} (1 - \beta \cdot p_j(t-1)) \quad \text{(from Eq. 3.5)} \\ &\geq 1 - \beta \cdot \sum_{j:neighbor\ of\ i} p_j(t-1) \end{aligned} \tag{3.16}$$

where the last step follows because all terms are positive.

Now, for $i = 1 \ldots N$,

$$\begin{aligned}
1 - p_i(t) &= (1 - p_i(t-1)) \cdot \zeta_i(t) + \delta \cdot p_i(t-1) \cdot \zeta_i(t) \\
&= (1 - (1-\delta) \cdot p_i(t-1)) \cdot \zeta_i(t) \\
&\geq (1 - (1-\delta) \cdot p_i(t-1)) \cdot \left(1 - \beta \sum_{j:neighbor\ of\ i} p_j(t-1)\right) \\
&\qquad \text{(using Eq. 3.16)} \\
&\geq (1 - (1-\delta) \cdot p_i(t-1)) \cdot \left(1 - \beta \sum_{j=1}^{N} \mathbf{A}_{j,i} \cdot p_j(t-1)\right) \\
&\qquad \text{(because } \mathbf{A}_{j,i} = 1 \text{ for neighbors only)} \\
&\geq 1 - (1-\delta) \cdot p_i(t-1) \\
&\quad -\beta \sum_j \mathbf{A}_{j,i} \cdot p_j(t-1) + \beta \cdot (1-\delta) \cdot p_i(t-1) \sum_j \mathbf{A}_{j,i} \cdot p_j(t-1) \\
&\geq 1 - (1-\delta) \cdot p_i(t-1) - \beta \sum_j \mathbf{A}_{j,i} \cdot p_j(t-1) \qquad (3.17)
\end{aligned}$$

No assumptions were required in this.

Thus,

$$p_i(t) \leq (1-\delta) \cdot p_i(t-1) + \beta \sum_j \mathbf{A}_{j,i} \cdot p_j(t-1) \qquad (3.18)$$

Writing this in vector form, we observe that this uses the same system matrix $\mathbf{S}$ from Equation 3.11:

$$\begin{aligned}
\vec{p}(t) &\leq \mathbf{S}\vec{p}(t-1) \quad \text{(using the definition of } \mathbf{S} \text{ from Eq. 3.11)} \\
&\leq \mathbf{S}^2\vec{p}(t-2) \leq \ldots \\
&\leq \mathbf{S}^t\vec{p}(0) \\
&\leq \sum_i {\lambda_{i,S}}^t \, \vec{u}_{i,S} \, \vec{u}_{i,S}{}' \, \vec{p}(0) \qquad (3.19)
\end{aligned}$$

where the last step is the spectral decomposition of $\mathbf{S}^t$. Using Eq. 3.12,

$$\begin{aligned}
\lambda_{i,S} &= 1 - \delta + \beta\lambda_{i,A} \\
&< 1 - \delta + \beta\frac{\delta}{\beta} \text{ (the sufficiency condition)} \\
&< 1 \\
\text{and so,} \quad {\lambda_{i,S}}^t &\approx 0 \quad \text{for all } i \text{ and large } t
\end{aligned}$$

Thus, the right-hand side of Eq. 3.19 goes to zero, implying that

$$\vec{p}(t) \rightsquigarrow 0 \text{ as } t \text{ increases}$$

implying that the infection dies out over time. □

Theorem 4 (Exponential Decay). *When an epidemic is diminishing (therefore $\beta/\delta < \frac{1}{\lambda_{1,A}}$), the probability of infection decays at least exponentially over time.*

Proof. We have:

$$\begin{aligned} \vec{p}(t) &\leq \sum_i {\lambda_{i,S}}^t \, \vec{u}_{i,S} \, \vec{u}_{i,S}{}' \, \vec{p}(0) \quad \text{(from Eq 3.19)} \\ &\leq {\lambda_{1,S}}^t * \mathbf{C} \end{aligned} \tag{3.20}$$

where $\mathbf{C}$ is a constant vector which depends on the entire spectrum of eigenvalues and eigenvectors. Since the value of $\lambda_{1,S}$ is less than 1 (because the epidemic is diminishing), the values of $p_i(t)$ are decreasing exponentially over time. The exact rate, however, depends on the entire spectrum of eigenvalues. □

Corollary 1. NLDS *subsumes the KW model for homogeneous or random Erdős-Rényi graphs.*

Proof. According to the KW model 3.1, the epidemic threshold in a random Erdős-Rényi graph is $\tau_{KW} = 1/\langle k \rangle$, where $\langle k \rangle$ is the average degree [Kephart and White, 1991]. It is easily shown that, in a homogeneous or random network, the largest eigenvalue of the adjacency matrix is $\langle k \rangle$. Therefore, our model yields the same threshold condition for random graphs, and thus, our *NLDS* model subsumes the KW model. □

Corollary 2. *The epidemic threshold τ_{NLDS} for a star topology, is exactly $\frac{1}{\sqrt{d}}$, where $\sqrt{d}$ is the square root of the degree of the central node.*

Proof. The eigenvalue of the adjacency matrix, λ_1, is simply $\sqrt{d}$. Thus, the epidemic threshold is $\tau_{NLDS} = \frac{1}{\sqrt{d}}$. □

Corollary 3. *The epidemic threshold for an infinite power-law network is zero. This matches results*

Proof. In a power-law network, the first eigenvalue of the adjacency matrix is $\lambda_{1,A} = \sqrt{d_{max}}$, where d_{max} is the maximum node degree in the graph [Mihail and Papadimitriou, 2002]. Since $d_{max} \propto ln(N)$ and N is infinite, $\lambda_{1,A}$ is infinite. Our epidemic threshold condition states that $\tau_{NLDS} = 1/\lambda_{1,A}$. Therefore, the epidemic threshold is effectively zero for infinite power-law networks. This result concurs with previous work, which found that infinite power-law networks lack epidemic thresholds [Pastor-Satorras and Vespignani, 2001]. □

Corollary 4. *Below the epidemic threshold (score $s < 1$), the expected number of infected nodes η_t at time t decays exponentially over time.*

Proof.

$$\begin{aligned} \eta_t &= \sum_{i=1}^{N} p_i(t) \\ &= \sum_i {\lambda_{1,S}}^t * C_i \quad \text{(from Theorem 4)} \\ &= {\lambda_{1,S}}^t * \sum_i C_i \end{aligned}$$

where C_i are the individual elements of the matrix $\mathbf{C}$ in Equation 3.20. Since $\sum_i C_i$ is a constant and $\lambda_{1,S} < 1$ (from Theorem 1), we see that n_t decays exponentially with time. □

3.6 Summary

Our work on viral propagation has two main contributions:

1. Given a graph, we found an epidemic threshold condition below which an infection dies out, but above which it may survive in the graph for a long time. Surprisingly, this epidemic threshold is found to depend on only *one* property of the graph: its largest eigenvalue λ_1. Specifically, the infection dies out when $\beta/\delta < 1/\lambda_1$. Thus, higher the value of λ_1, higher the susceptibility of the graph to viral outbreaks. This result is an important design consideration not only in the development of virus-resistant computer networks, but also in understanding the spread of rumors, or fads, or the effectiveness of information dissemination programs.

2. In addition, we proposed a very general dynamical-systems framework, which can now be used in other similar cases where propagation depends on network topology. In fact, in Chapter 4, we will demonstrate the applicability of this methodology in calculating the "survivability" of information in sensor networks.

An important avenue of future work is using this threshold condition in developing good immunization schemes, to maximally disrupt the spread of a virus.

Chapter 4

Information survival in sensor and P2P networks

"Consider a piece of information being spread within a sensor or P2P network with failing links and nodes. What conditions on network properties determine if the information will survive in the network for long, or die out quickly?

Sensor and Peer-to-peer (P2P) networks have recently been employed in a wide range of applications:

- *Oceanography:* Video sensors have been used to generate panoramic views of the near-shore ocean surface [Holman et al., 2003].
- *Infrastructure monitoring:* Software sensors have been used to monitor and collect statistics on various bottleneck points in computers, such as CPU load, network bandwidth, and so on.
- *Parking Space tracking:* This, and several similar applications, have been developed and deployed under the IrisNet framework [Gibbons et al., 2003].

A lot of work has also been done recently on collating information from many different sensors and answering user queries efficiently [Madden et al., 2003, Garofalakis and Kumar, 2004, Considine et al., 2004, Nath et al., 2004].

We look at the problem of survivability of information in a sensor or P2P network under node and link failures. For example, consider a sensor network where the communication between nodes is subject to loss (link failures), and sensors may fail (node failures). In such networks, we may want to maintain some static piece of information, or "datum", which, for the sake of exposition, we refer to as "Smith's salary". If only one sensor node keeps Smith's salary, that node will probably fail sometime and the information will be lost. To counter this, nodes that have the datum can broadcast it to other nodes, spreading it through the network and increasing its chances of survival in the event of node or link failures.

Informally, the problem we solve is:

PROBLEM: *Under what conditions can we expect Smith's salary to survive in the sensor network?*

This question is similar to the one we asked for viral propagation, and in order to answer it, we must surmount the same issues. Although the problem definition is deceptively simple, there is no known practical, exact solution. The obvious one, discussed later in Section 4.2, requires Markov Chains, and is prohibitively expensive: its cost is proportional to 3^N, where N is the number of nodes. For a network of $N \approx 200$ nodes, the effort is comparable to the number of electrons in the universe.

As in the analysis of viral propagation, we will use the dynamical systems framework to solve the current problem. We will find a threshold below which the information is expected to die out quickly, but above which the information may survive. This threshold, once again, depends only on the largest eigenvalue of a appropriately constructed square matrix.

The layout of this chapter is as follows: Section 4.1 discusses some background work related to the problem. In Section 4.2 gives a precise problem definition, and our results on the information survival threshold are described in Section 4.3. Section 4.4 details experiments showing the accuracy of our predictions. We provide details of the proofs in Section 4.5, followed by a summary in Section 4.6.

4.1 Related Work

Much of the applicable related work has already been described in Section 3.1. Here, after a brief recap of those ideas, we will focus on prior work in sensor networking.

4.1.1 Recap: Propagation over a network

The key concepts were:

- *The SIS and SIR models:* These describe the behavior of a single node during viral propagation. A susceptible (S) node can be attacked and infected by an infected neighbor. An infected (I) node can be cured locally, in which case it can either become susceptible again (in the SIS model), or can be removed (R) and achieve immunity (in the SIR model). The spread of information in a sensor network is similar to the SIS model: a node can get the information from a neighbor, but it can fail and lose information. When it is replaced, it becomes "susceptible" again.

- *The epidemic threshold:* For the SIS model, we found a closed-form formula for the epidemic threshold, below which a viral outbreak is expected to die out exponentially quickly, but above which it may survive for a long time (see Chapter 3). This formula is surprisingly simple: it depends only on the "birth" and "death" rates of the virus, and the *largest eigenvalue of the network topology*.

4.1.2 Sensor Networks and Gossip-based Protocols

Graphs and sensor networks have attracted a lot of interest lately, for quick and efficient aggregation of information [Garofalakis and Kumar, 2004, Considine et al., 2004], for understanding "trust" and "distrust" in online social networks [Guha et al., 2004], and in several other areas.

Large sensor and P2P networks typically have dynamic node populations with a high rate of node turnover (called high *churn* [Rhea et al., 2004]), and "gossip" protocols have recently been used to handle such situations. These protocols spread information using a (possibly random) network of connections between nodes, as in our case, and have proved useful in reliable multicast and broadcast [Levis et al., 2004, Gupta et al., 2002, Luo et al., 2003, Birman et al., 1999], resource location [Kempe et al., 2001, Renesse, 2000], failure detection [Wang and Kuo, 2003, Renesse et al., 1998, Sistla et al., 2001, Burns et al., 1999, Zhuang et al., 2005], database aggregation [Kempe et al., 2003], database and peer-to-peer replication [Demers et al., 1987], and ensuring the stability of dynamic hash table-based peer-to-peer systems [Gupta et al., 2003, Rhea et al., 2004]. Several empirical and theoretical studies of the rates of propagation and convergence of gossip protocols have also been made [Boyd et al., Ganesan et al., 2002, Levis et al., 2004, Kempe et al., 2001, Kempe and Kleinberg, 2002].

However, they all assume that the initial infection or rebroadcast rate is high enough that dying out is not a concern. With the rise of sensor and peer to peer networks characterized by high churn, theory that describes the survivability of data in such networks is increasingly important, and that is the central problem we address.

4.2 Model for Information Propagation

As in Chapter 3, we have a sensor/P2P/social network of N nodes (sensors or computers or people) and E directed links between them. Our analysis assumes very small discrete timesteps of size Δt, where $\Delta t \to 0$. Within a Δt time interval, each node i has a probability r_i of trying to broadcast its information every timestep, and each link $i \to j$ has a probability β_{ij} of being "up", and thus correctly propagating the information to node j. Each node i also has a node failure probability $\delta_i > 0$ (e.g., due to battery failure in sensors). Every dead node j has a rate γ_j of returning to the "up" state, but without any information in its memory (e.g., due to the periodic replacement of dead batteries). These and other symbols are listed in Table 4.1.

The system is a Markov chain with 3^N states; each state corresponds to one possible network configuration, with each of the N nodes being in one of three states: "Has Info," "No Info," or "Dead." There is a set of "absorbing" states (where no node "Has Info"), which can be reached from any starting state. Thus, the information will die out with probability 1. However, from a practical point of view, in some parameter combinations this extinction happens quickly, while in others it is expected to be longer than the age of the universe (for large graphs) - in the latter cases, the datum practically 'survives'.

Definition 13 (Fast Extinction). *"Fast extinction" is the setting where the number $\bar{C}(t)$ of "carriers" (i.e., nodes in "Has Info" state) decays exponentially over time ($\bar{C}(t) \propto c^{-t}, c > 1$).*

Now, we formally state our problem:

Symbol	Description
N	Number of nodes in the network
β_{ij}	*Link quality:* Probability that link $i \rightarrow j$ is up
δ_i	*Death rate:* Probability that node i dies
γ_i	*Resurrection rate:* Probability that node i comes back up
r_i	*Retransmission rate:* Probability that node i broadcasts
$p_i(t)$	Probability that node i is alive at time t and has info
$q_i(t)$	Probability that node i is alive at time t but without info
$1 - p_i(t) - q_i(t)$	Probability that node i is dead
$\zeta_i(t)$	Probability that node i does *not* receive info from *any* of its neighbors at time t
$\vec{p}(t)$, $\vec{q}(t)$	Probability column vectors: $\vec{p}(t) = (p_1(t), p_2(t), \ldots, p_N(t))'$, $\vec{q}(t) = (q_1(t), q_2(t), \ldots, q_N(t))'$
f: $\mathcal{R}^{2N} \rightarrow \mathcal{R}^{2N}$	Function representing a dynamical system
$\nabla(\mathbf{f})$	The Jacobian matrix of f(.)
$\mathbf{S}$	The $N \times N$ "system" matrix
$\lambda_{\mathbf{S}}$	An eigenvalue of the $\mathbf{S}$ matrix
$\lambda_{1,\mathbf{S}}$	The largest eigenvalue (in magnitude) of $\mathbf{S}$
$s = \lvert\lambda_{1,\mathbf{S}}\rvert$	"Survivability score" = Magnitude of $\lambda_{1,\mathbf{S}}$

Table 4.1: *Table of symbols*

PROBLEM:

- *Given: the network topology (link "up" probabilities)* β_{ij}*, the retransmission rates* r_i*, the resurrection rates* γ_i *and the death rates* δ_i *(*$i = 1 \ldots N$*,* $j = 1 \ldots N$*)*
- *Find the condition under which a datum will suffer "fast extinction."*

To simplify the problem and to avoid dependencies on starting conditions, we consider the case where all nodes are initially in the "Has Info" state.

As in analysis of viral propagation (Chapter 3), we will convert this into a dynamical system, and answer questions in that system. Let the probability of node i being in the "Has Info" and "No Info" states at time-step t be $p_i(t)$ and $q_i(t)$ respectively. Thus, the probability of its being dead is $(1 - p_i(t) - q_i(t))$. Starting from state "No Info" at time-step $t - 1$, node i can acquire this information (and move to state "Has Info") if it receives a communication from some other node j. Let $\zeta_i(t)$ be the probability that node i does *not* receive the information from *any* of its neighbors. Then, assuming the neighbors' states are independent:

$$\zeta_i(t) = \Pi_{j=1}^{N} \left(1 - r_j \beta_{ji} p_j(t-1)\right) \tag{4.1}$$

For each node i, we can use the transition matrix in Figure 4.1 to write down the probabilities of being in each state at time t, *given* the probabilities at time $t - 1$ (recall that we use very small

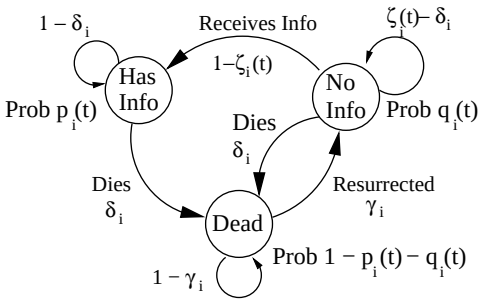


Figure 4.1: *Transitions for each node:* This shows the three states for each node, and the probabilities of transitions between states.

timesteps Δt, and so we can neglect second-order terms). Thus:

$$\begin{aligned} p_i(t) &= p_i(t-1)\,(1-\delta_i) \\ &\quad +q_i(t-1)\,(1-\zeta_i(t)) \qquad (4.2) \\ q_i(t) &= q_i(t-1)\,(\zeta_i(t)-\delta_i) \\ &\quad +(1-p_i(t-1)-q_i(t-1))\,\gamma_i \qquad (4.3) \end{aligned}$$

From now on, we will only work on this dynamical system. Specifically, we want to find the condition for fast extinction under this system.

4.3 Information Survival Threshold

Define $\mathbf{S}$ to be the $N \times N$ *system matrix*:

$$\mathbf{S}_{ij} = \begin{cases} 1-\delta_i & \text{if } i=j \\ r_j \beta_{ji} \dfrac{\gamma_i}{\gamma_i+\delta_i} & \text{otherwise} \end{cases} \qquad (4.4)$$

Let $|\lambda_{1,\mathbf{S}}|$ be the magnitude of the largest eigenvalue (in magnitude). Let $\hat{C}(t)$ to be the expected number of carriers at time t according to this dynamical system; $\hat{C}(t) = \sum_{i=1}^{N} p_i(t)$.

Theorem 5 (Condition for fast extinction). *Define $s = |\lambda_{1,\mathbf{S}}|$ to be the "survivability score" for the system. If*

$$\boxed{s = |\lambda_{1,\mathbf{S}}| < 1}$$

then we have fast extinction in the dynamical system, that is, $\hat{C}(t)$ decays exponentially quickly over time.

Proof. The theorem is proved later in Section 4.5. □

Definition 14 (Threshold). *We will use the term "below threshold" when $s < 1$, "above threshold" when $s > 1$, and "at the threshold" for $s = 1$.*

Corollary 5 (Homogeneous case). *If $\delta_i = \delta, r_i = r, \gamma_i = \gamma$ for all i, and $\mathbf{B} = [\beta_{ij}]$ is a symmetric binary matrix (links are undirected, and are always up or always down), then the condition for fast extinction is:*

$$\boxed{\frac{\gamma r}{\delta(\gamma+\delta)}\lambda_{1,\mathbf{B}} < 1}$$

One way to visualize this is to think of the spread of information as a random walk on the sensor network. The information "traverses one hop" according to the probabilities from the $\mathbf{B}$ matrix, and thus "traverses n hops" according to $\mathbf{B}^n$. But $\mathbf{B}^n$ grows as $\lambda^n_{1,\mathbf{B}}$ (indeed, this is the basis of the "power-method" for finding the largest eigenvalue of a matrix). This growth is increased by higher retransmission rate r, but the information can die out with probability δ, which reduces the spread. Also, a "target" node is not "Dead" only $\frac{\gamma}{\gamma+\delta}$ fraction of the time, so the information can spread only so fast. Thus, the survivability score $\frac{\gamma r}{\delta(\gamma+\delta)}\lambda_{1,\mathbf{B}}$ represents the rate of spread, taking all these effects into account. If this is less than one, the information is not spreading quickly enough, and becomes extinct quickly.

Theorem 5 has several corollaries and special cases, which are of considerable interest. For clarity, we only consider homogeneous undirected graphs here (as in Corollary 5), and proofs can be obtained by application of the Theorem.

Corollary 6. *Consider two scenarios on the same network with all parameters the same except for varying rates γ_1 and γ_2 for dead nodes coming "up" again. Suppose $\gamma_1 > \gamma_2$. Then, the first system has higher survivability score.*

Proof. $s_1 = \frac{\gamma_1 r}{\delta(\gamma_1+\delta)}\lambda_{1,\mathbf{B}} = \frac{r}{\delta\left(1+\frac{\delta}{\gamma_1}\right)}\lambda_{1,\mathbf{B}} > \frac{r}{\delta\left(1+\frac{\delta}{\gamma_2}\right)}\lambda_{1,\mathbf{B}} = \frac{\gamma_2 r}{\delta(\gamma_2+\delta)}\lambda_{1,\mathbf{B}} = s_2.$ □

Corollary 7. *Our model subsumes the SIS model of viral infection as a special case.*

Proof. The SIS model consists has only two states per node: "Has Infection" and "No Infection." In our model, we can increase γ so that a "dead" node comes back "up" very quickly, giving the appearance of only *two* states: "Has Info" and "No Info". In this situation, γ is large ($\gamma \approx 1$) but δ_i is small (due to small timesteps). From Corollary 5, the fast-extinction condition becomes $\delta\lambda_{1,\mathbf{B}} < 1$ (recall that the matrix $B = [\beta_{ij}]$). This is exactly the epidemic threshold condition for the SIS model [Wang et al., 2003, Ganesh et al., 2005]. □

Corollary 8. *If the "dead" nodes are never replaced ($\gamma = 0$), or if the nodes do not transmit at all ($r = 0$), we* always *have fast extinction.*

Proof. $s = |\lambda_{1,\mathbf{S}}| = \frac{\gamma r}{\delta(\gamma+\delta)}\lambda_{1,\mathbf{B}} = 0$ for $\gamma = 0$ or $r = 0$. Hence, we always have fast extinction. □

Corollary 9. *Consider two networks with link "up" probabilities given by matrices $\mathbf{B}_1$ and $\mathbf{B}_2$. If $|\lambda_{1,\mathbf{B}_1}| > |\lambda_{1,\mathbf{B}_2}|$, then the first network has higher survivability score.*

Proof. $s_1 = \frac{\gamma r}{\delta(\gamma+\delta)}\lambda_{1,\mathbf{B}_1} > \frac{\gamma r}{\delta(\gamma+\delta)}\lambda_{1,\mathbf{B}_2} = s_2.$ □

Corollary 10 (P2P resilience). *A "star" network has higher survivability score than a "ring" network with the same number of nodes.*

Proof. $|\lambda_{1,\mathbf{B}}|_{star} = \sqrt{N-1} > 2 = |\lambda_{1,\mathbf{B}}|_{ring}$. From Corollary 9, the "star" network has higher survivability score. □

4.4 Experiments

We want to answer the following questions:

(Q1) How close is our approximation technique to the "truth", for both real and synthetic graphs?

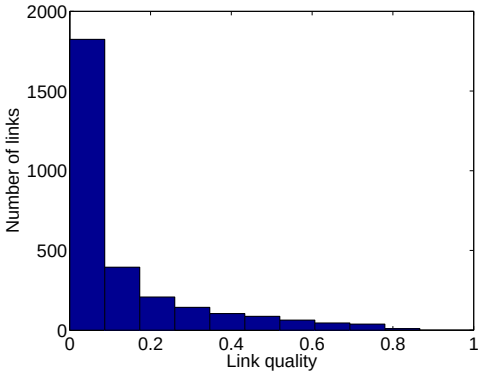
(Q2) How accurate is our threshold condition (Theorem 5)?

The datasets used were:

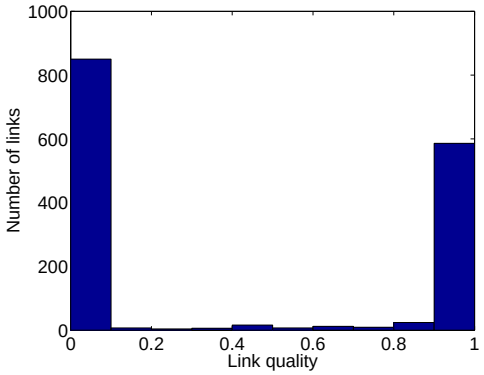
- *GRID*: This is a large synthetic 2D grid. Link "up" probabilities (β_{ij}) are fixed at 0.1 between all neighbors on the grid.
- *GNUTELLA*: This is a snapshot of the Gnutella peer-to-peer file sharing network, collected in March 2001 [Ripeanu et al., 2002]. Link "up" probabilities β_{ij} are set to 0.1 for existing edges.
- *INTEL*: This is a 54-node sensor network observed over a period of 33 days [Intel]. Link probabilities were derived from the collected data. The nodes were deployed in a lab with a rectangular shape and "soft" walls which can be penetrated by radio signals (see Figure 4.3(a)), leading to high average node degree (≈ 46) in the network. The link qualities β_{ij} are smeared-out over the entire range, as shown in Figure 4.2(a). The average link quality (considering only the links with non-zero link quality) is very low (0.14).
- *MIT*: This is a 40-node sensor network at MIT (see [Hull et al., 2004] for an earlier version of the network). Figure 4.3(b) shows the placement of sensors. The central region blocks radio signals, creating an elongated "corridor" of sensor communications around it. This implies low average node degree (≈ 18); however, the link quality distribution is very peaked, as shown in Figure 4.2(b). This leads to a high value of 0.92 for the average link quality (again only considering the non-zero quality links). Note that these conditions are the exact opposite of what we see for the *INTEL* dataset.

4.4.1 (Q1) Accuracy of the dynamical system

For each of the datasets, the parameters were set so that the system was below, above and at the threshold (that is, the survivability score s was less then, equal to or greater than 1) according to Theorem 5. Table 4.2 shows these parameter values. All nodes initially carry the information. The simulation is then run according to the state diagram shown in Figure 4.1 for $10,000$ steps, and

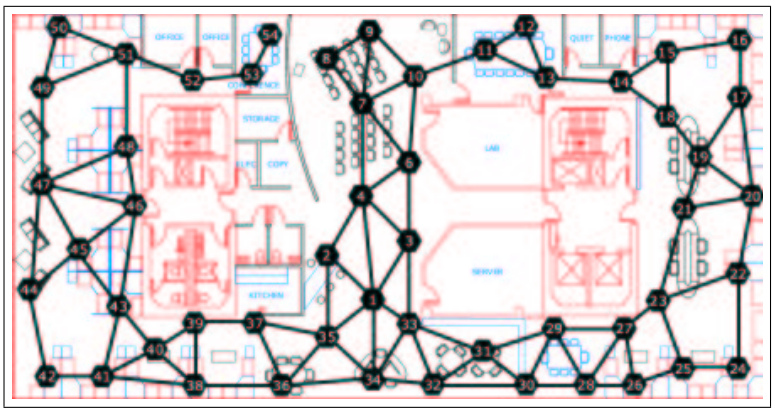


(a) *INTEL* link qualities



(a) *MIT* link qualities

Figure 4.2: *Link quality distributions:* Plots (a) and (b) plots the number of links versus link quality. Pairs of sensors which cannot communicate with each other have a link quality of 0. While the *INTEL* distribution shows a broad range of link qualities, the *MIT* distribution is very highly peaked.



(a) *INTEL* sensor map

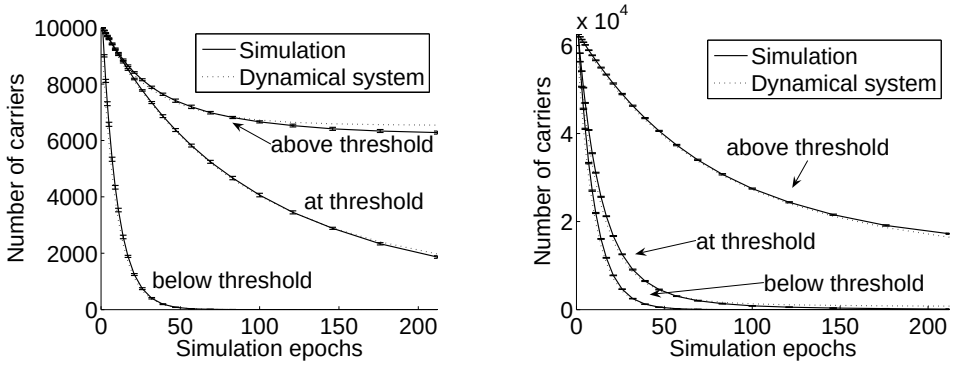
(b) *MIT* sensor map

Figure 4.3: *Sensor placement maps:* Black numbered dots for *INTEL*, and black dots for *MIT*.

the number of carriers (nodes with information) over the epochs of simulation was recorded. Each simulation was repeated 100 times.

Figure 4.4 shows the number of carriers over time (only 200 simulation epochs are shown for ease of visualization; the results are similar over $10,000$ timesteps). Each dataset is investigated under three scenarios: above, at and below our threshold condition. For each scenario, we show the simulation result in solid lines, along with its confidence interval (one standard deviation). While the confidence intervals are very small for *GRID* and *GNUTELLA*, the *INTEL* and *MIT* datasets show wider confidence intervals: this is due to their small sizes. In addition to the simulations, we ran our dynamical system (Equations 4.2-4.3) using exactly the same parameters; the number of carriers in this case is shown in dotted lines. We make the following observations:

- *The dynamical system is very accurate:* The dotted lines of our dynamical system are almost indistinguishable from the solid lines of the simulation (relative error is just around 1%).



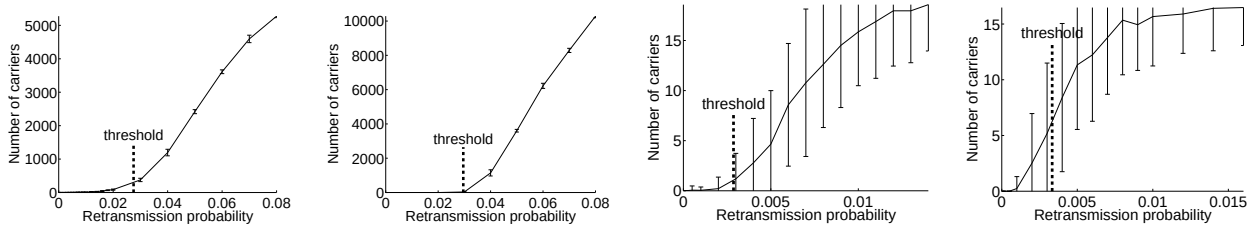
(a) *GRID* (b) *GNUTELLA*

(c) *INTEL* (d) *MIT*

Figure 4.4: *Number of carriers versus time (simulation epochs):* Each plot shows the evolution of the dynamical system (dotted lines) and the simulation (solid lines). Confidence intervals are shown for the simulation results. Three cases are shown for each dataset: below threshold, at the threshold, and above the threshold, with parameter settings as shown in Table 4.2. There are two observations: (1) The dynamical system (dotted lines) is very close to the simulations (solid lines), demonstrating the accuracy of Equations 4.2-4.3. (2) Also, the number of carriers dies out very quickly below the threshold, while the information "survives" above the threshold.

Dataset	w.r.t. threshold	δ	γ	r	Survivability s
GRID	below	0.1	0.01	0.1	0.90
	at	0.01	0.004	0.1	1.001
	above	0.01	0.1	0.1	1.02
GNUTELLA	below	0.1	0.01	0.1	0.91
	at	0.07	0.004	0.1	1.003
	above	0.01	0.01	0.1	1.05
INTEL	below	0.1	0.01	0.1	0.96
	at	0.02	0.0006	0.1	1.0003
	above	0.01	0.01	0.1	1.33
MIT	below	0.15	0.01	0.1	0.96
	at	0.05	0.0006	0.1	1.01
	above	0.01	0.01	0.1	1.88

Table 4.2: *Parameter settings for the datasets.*



(a) *GRID* (b) *GNUTELLA* (c) *INTEL* (d) *MIT*

Figure 4.5: *Number of carriers after a "long" time, versus the retransmission probability:* The dashed vertical line shows our threshold ($s = 1$). The information dies out below our threshold, but survives above it.

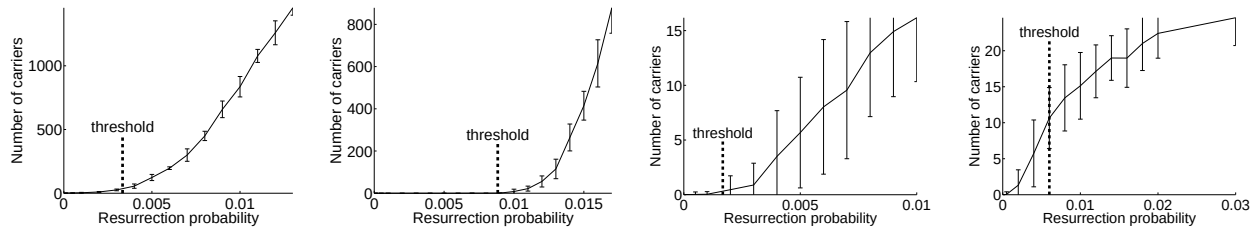
Thus, equations 4.2-4.3 are highly accurate for a wide variety of real-world scenarios.

- *The information dies out below our threshold:* For all the datasets, the number of carriers goes to zero very quickly below the threshold. Thus, the information does not survive.

- *Above our threshold, the number of carriers practically stabilizes:* In other words, the information survives for a "long" time.

4.4.2 (Q2) Accuracy of the threshold condition

In this set of experiments, we modify one parameter while keeping all the others fixed. The link qualities β_{ij} depend on the environment, while the death rate δ is intrinsic to the sensor and its battery; thus, we only perform experiments that modify the retransmission rate r and the resurrection rate γ. For each dataset, we run simulations for many values of r and γ, and count the number of carriers left after a "long" time ($1,000$ simulation epochs). For each setting, 100 simulation runs are performed to provide confidence intervals.



(a) *GRID* (b) *GNUTELLA* (c) *INTEL* (d) *MIT*

Figure 4.6: *Number of carriers after a "long" time, versus the resurrection probability:* The dashed vertical line shows our threshold. Again, our threshold is very accurate.

Varying retransmission rate r

For the purpose of this experiment, the death rate δ and the resurrection rate γ were set to 0.01. Figure 4.5 shows the number of carriers left after a "long" time, versus different values of the retransmission rate r, on all four datasets. The dashed vertical line marks the value of r which leads to a survivability score of $s = 1$ according to Theorem 5, that is, the "at-the-threshold" scenario. *INTEL* and *MIT* have higher variance (and hence wider confidence intervals) due to their small sizes. We observe the following:

- *Below our threshold, the information has died out:* The number of carriers is either zero or very close to zero for all the datasets. This is exactly in accordance with Theorem 5.
- *Above the threshold, the information survives:* Even after a "long" time, there is a significant population of carriers in the network.
- *Thus, the threshold condition is very accurate.*

Varying resurrection rate γ

Now, the resurrection rate γ is varied, while keeping the retransmission rate r fixed at 0.1 and the death rate δ at 0.01. Figure 4.6 shows the results. The dashed vertical line shows our threshold value. Once again, we can see that for all the datasets, the information dies out below the threshold, but survives above the threshold. Thus, our threshold is accurate.

4.5 Details of proofs

Theorem 2 (Condition for fast extinction). *Define $s = |\lambda_{1,\mathbf{S}}|$ to be the "survivability score" for the system. If*

$$\boxed{s = |\lambda_{1,\mathbf{S}}| < 1}$$

then we have fast extinction in the dynamical system, that is, $\hat{C}(t)$ decays exponentially quickly over time.

Proof. The theorem follows from Lemma 3 and Theorems 3 and 4 below. We will first show that the scenario with no information survival ($p_i(t) = 0$) forms a fixed point of the dynamical system. Then, we show that when $s < 1$, this fixed point is *asymptotically stable* under small perturbations (this is how we derived the condition in this theorem). Finally, we show that our threshold is insensitive to the starting state: when $s < 1$, $p_i(t) \to 0$ and thus $\hat{C}(t) \to 0$ exponentially quickly. □

Definition 15 (Asymptotic Stability of a Fixed Point). *A fixed point P_f is "asymptotically stable" if, on a slight perturbation from P_f, the system returns to P_f (as against moving away, or staying in the neighborhood of P_f but not approaching it) [Hirsch and Smale, 1974].*

Lemma 3 (Fixed Point). *The values $\left(p_i(t) = 0, q_i(t) = \frac{\gamma_i}{\gamma_i + \delta_i}\right)$ for all nodes i, are a fixed point of Equations 4.2-4.3. Proved by a simple application of the Equations.*

Theorem 3 (Stability of the fixed point). *The fixed point of Lemma 3 is* asymptotically stable *if the system is below the threshold, that is, $s = |\lambda_{1,\mathbf{S}}| < 1$.*

Proof. First, we define the column vector $\vec{p}(t) = (p_1(t), p_2(t), \ldots, p_N(t))'$. Define $\vec{q}(t)$ similarly. Let $\vec{v}(t) = (\vec{p}(t), \vec{q}(t))$ be the concatenation of these two vectors, and $\vec{v}_f$ be the vector $\vec{v}(t)$ at the fixed point defined in Lemma 3. Then, the entire system can be described as:

$$\begin{aligned} \vec{v}(t) &= f\left(\vec{v}(t-1)\right) \quad \text{where} \\ f_i\left(\vec{v}(t-1)\right) &= \begin{cases} p_i(t-1)\left(1-\delta_i\right) & \\ \quad + q_i(t-1)\left(1-\zeta_i(t)\right) & \text{if } i \le N \\ q_i(t-1)\left(\zeta_i(t)-\delta_i\right) & \\ \quad + \left(1 - p_i(t-1) - q_i(t-1)\right)\gamma_i & \text{if } i > N \end{cases} \end{aligned} \tag{4.5}$$

where $p_i(t-1)$ and $q_i(t-1)$ are the corresponding entries in $\vec{v}(t-1)$.

To check for the asymptotic stability of a fixed point, we can use the following condition.

Lemma 4. *(From [Hirsch and Smale, 1974]) Define $\nabla(\mathbf{f})$ (also called the Jacobian matrix) to be a $2N \times 2N$ matrix such that*

$$[\nabla(\mathbf{f})]_{ij} = \frac{\partial f_i\left(\vec{v}(t-1)\right)}{\partial \vec{v}_j(t-1)} \tag{4.6}$$

Then, if the largest eigenvalue (in magnitude) of $\nabla(\mathbf{f})$ at $\vec{v}_f$ (written $\nabla(\mathbf{f})|_{\vec{v}_f}$) is less than 1 in magnitude, the system is asymptotically stable at $\vec{v}_f$. Also, if f is linear and the condition holds, then the dynamical system will exponentially tend to the fixed point irrespective of initial state.

Applying this to our dynamical system (Equation 4.5), we get a block-matrix form for $\nabla(\mathbf{f})|_{\vec{v}_f}$:

$$\nabla(\mathbf{f})|_{\vec{v}_f} = \left[\begin{array}{c|c} \mathbf{S} & 0 \\ \hline \mathbf{S_1} & \mathbf{S_2} \end{array}\right] \tag{4.7}$$

Here, all of $\mathbf{S}$, $\mathbf{S_1}$ and $\mathbf{S_2}$ are $N \times N$ matrices, defined as:

$$\mathbf{S}_{ij} = \begin{cases} 1-\delta_i & \text{if } i=j \\ r_j\beta_{ji}\frac{\gamma_i}{\gamma_i+\delta_i} & \text{otherwise} \end{cases} \quad \text{as in Eqn. 4.4}$$

$$\mathbf{S_1}_{ij} = \begin{cases} -\gamma_i & \text{if } i=j \\ -r_j\beta_{ji}\frac{\gamma_i}{\gamma_i+\delta_i} & \text{otherwise} \end{cases} \tag{4.8}$$

$$\mathbf{S_2}_{ij} = \begin{cases} 1-\gamma_i-\delta_i & \text{if } i=j \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

In order to check for asymptotic stability (Lemma 4), we must find the largest eigenvalues (in magnitude) of $\nabla(\mathbf{f})|_{\vec{v}_f}$. Let $\vec{x}$ be an eigenvector (of $2N$ elements) of $\nabla(\mathbf{f})|_{\vec{v}_f}$, with the form

$$\vec{x} = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \end{bmatrix}$$

where both $\vec{x}_1$ and $\vec{x}_2$ are vectors of length N. Thus,

$$\nabla(\mathbf{f})|_{\vec{v}_f}\vec{x} = \left[\begin{array}{c|c} \mathbf{S} & 0 \\ \hline \mathbf{S_1} & \mathbf{S_2} \end{array}\right] \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \end{bmatrix} = \lambda_{\nabla(\mathbf{f})|_{\vec{v}_f}} \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \end{bmatrix}$$

$$\text{which implies, } \mathbf{S}\vec{x}_1 = \lambda_{\nabla(\mathbf{f})|_{\vec{v}_f}}\vec{x}_1 \tag{4.10}$$

$$\text{and, } \mathbf{S_1}\vec{x}_1 + \mathbf{S_2}\vec{x}_2 = \lambda_{\nabla(\mathbf{f})|_{\vec{v}_f}}\vec{x}_2 \tag{4.11}$$

The only possible solutions are:

- *Case* 1*:* $\lambda_{\nabla(\mathbf{f})|_{\vec{v}_f}}$ *is an eigenvalue of* $\mathbf{S}$*:* This is implied by Equation 4.10.
- *Case* 2*:* $(1-\gamma_i-\delta_i)$ *is an eigenvalue of* $\mathbf{S}$*, for all* i: This happens if $\vec{x}_1 = \vec{0}$, and follows from Equation 4.11.

The largest eigenvalue from Case 2 is always less than 1 in magnitude (since $\delta_i > 0$). The largest eigenvalue (in magnitude) from Case 1 is exactly $\lambda_{1,\mathbf{S}}$. Thus, if $|\lambda_{1,\mathbf{S}}| < 1$, then from Lemma 4, the system is asymptotically stable. □

Theorem 4 (Insensitivity to the starting state). *If* $|\lambda_{1,\mathbf{S}}| < 1$*, then* $p_i(t) \to 0$ *exponentially quickly for all nodes* i*, irrespective of the starting state.*

Proof. We will prove this in three parts:

1. $p_i(t) + q_i(t) \to \frac{\gamma_i}{\gamma_i+\delta_i}$ exponentially quickly for all i. Since both $p_i(t)$ and $q_i(t)$ are non-negative (they are probabilities), this implies that either (1) $q_i(t) \to \frac{\gamma_i}{\gamma_i+\delta_i}$ and thus $p_i(t) \to 0$, OR (2) $q_i(t) < \frac{\gamma_i}{\gamma_i+\delta_i}$. One of these two situations occurs exponentially quickly.

2. If $q_i(T) < \frac{\gamma_i}{\gamma_i+\delta_i}$ at any time $t = T$, then $q_i(t) < \frac{\gamma_i}{\gamma_i+\delta_i}$ for all $t > T$.

3. Under the condition that $q_i(t) < \frac{\gamma_i}{\gamma_i+\delta_i}$ and $|\lambda_{1,\mathbf{S}}| < 1$, the dynamical system of Equations 4.2-4.3 tends to the fixed point $\left(p_i(t) = 0, q_i(t) = \frac{\gamma_i}{\gamma_i+\delta_i}\right)$ exponentially quickly.

Thus, when $|\lambda_{1,\mathbf{S}}| < 1$, we always end up with $p_i(t) \to 0$ for all i, and this happens exponentially quickly. So, the expected number of nodes with information $\hat{C}(t) = \sum_{i=1}^{N} p_i(t) \to 0$ exponentially quickly. All these three parts are proven below. □

Lemma 5. *$p_i(t) + q_i(t) \to \frac{\gamma_i}{\gamma_i+\delta_i}$ exponentially quickly for all i.*

Proof. From Equations 4.2 and 4.3, we can get:

$$\begin{aligned} 1 - p_i(t) - q_i(t) &= p_i(t-1)\delta_i + q_i(t-1)\delta_i \\ &+ (1 - p_i(t-1) - q_i(t-1))(1 - \gamma_i) \end{aligned} \tag{4.12}$$

Let $x_i(t) = p_i(t) + q_i(t)$. Then, from Equation 4.12,

$$\begin{aligned} 1 - x_i(t) &= x_i(t-1)\delta_i \\ &\quad +(1 - x_i(t-1))(1 - \gamma_i) \\ \text{and so, } x_i(t) &= \gamma_i + x_i(t-1)(1 - \gamma_i - \delta_i) \end{aligned} \tag{4.13}$$

Equation 4.13 is a *linear* dynamical system. A simple application of Lemma 4 shows that the system converges to the only fixed point of

$$\begin{aligned} x_i(t) &= \frac{\gamma_i}{\gamma_i + \delta_i} \\ \text{that is, } p_i(t) + q_i(t) &= \frac{\gamma_i}{\gamma_i + \delta_i} \end{aligned} \tag{4.14}$$

Again, by Lemma 4, this convergence is exponentially quick. □

Lemma 6. *If $q_i(T) < \frac{\gamma_i}{\gamma_i+\delta_i}$ at any time $t = T$, then $q_i(t) < \frac{\gamma_i}{\gamma_i+\delta_i}$ for all $t > T$.*

Proof. We prove this by induction.
Base Case: $q_i(T) < \frac{\gamma_i}{\gamma_i+\delta_i}$
Assumption: Suppose $q_i(t-1) < \frac{\gamma_i}{\gamma_i+\delta_i}$ at some time instant $t - 1$.

Induction: Applying Equation 4.3,

$$\begin{aligned} q_i(t) &= q_i(t-1)\left(\zeta_i(t)-\delta_i\right) \\ &\quad + \left(1-p_i(t-1)-q_i(t-1)\right)\gamma_i \\ &\leq q_i(t-1)(1-\delta_i) + \left(1-q_i(t-1)\right)\gamma_i \\ &\qquad \text{(since } \zeta_i(t)\leq 1 \text{ and } p_i(t-1)\geq 0\text{)} \\ &\leq \gamma_i + q_i(t-1)\left(1-\gamma_i-\delta_i\right) \\ &< \gamma_i + \frac{\gamma_i}{\gamma_i+\delta_i}\left(1-\gamma_i-\delta_i\right) \\ &\qquad \text{(from the Assumption)} \\ &< \frac{\gamma_i}{\gamma_i+\delta_i} \end{aligned}$$

Thus, $q_i(t) < \frac{\gamma_i}{\gamma_i+\delta_i}$ for all time $t > T$. □

Lemma 7. *Under the condition that* $q_i(t) < \frac{\gamma_i}{\gamma_i+\delta_i}$ *and* $|\lambda_{1,\mathbf{S}}| < 1$*, the dynamical system of Equations 4.2-4.3 tends to the fixed point* $\left(p_i(t)=0, q_i(t)=\frac{\gamma_i}{\gamma_i+\delta_i}\right)$ *exponentially quickly.*

Proof. We can use $q_i(t) < \frac{\gamma_i}{\gamma_i+\delta_i}$ in Equation 4.2 to get:

$$p_i(t) < p_i(t-1)\left(1-\delta_i\right) + \frac{\gamma_i}{\gamma_i+\delta_i}\left(1-\zeta_i(t)\right) \tag{4.15}$$

Now, we have the following inequality regarding $\zeta_i(t)$:

$$\begin{aligned} \zeta_i(t) &= \Pi_{j=1}^{N}\left(1-r_j\beta_{ji}p_j(t-1)\right) \\ &\geq 1-\Sigma_{j=1}^{N} r_j\beta_{ji}p_j(t-1) \\ \text{and thus, } \left(1-\zeta_i(t)\right) &\leq \Sigma_{j=1}^{N} r_j\beta_{ji}p_j(t-1) \end{aligned} \tag{4.16}$$

Using Equation 4.16 in 4.15:

$$0 \leq p_i(t) < p_i(t-1)\left(1-\delta_i\right) + \frac{\gamma_i}{\gamma_i+\delta_i}\Sigma_{j=1}^{N} r_j\beta_{ji}p_j(t-1)$$

Converting to the matrix form, we see

$$\begin{aligned} \vec{0} \leq \vec{p}(t) &< \mathbf{S}\vec{p}(t-1) \quad \text{(from Equation 4.4)} \\ &< \mathbf{S}^2\vec{p}(t-2) < \ldots \\ &< \mathbf{S}^t\vec{p}_0 \end{aligned} \tag{4.17}$$

What is $\mathbf{S}^t\vec{p}_0$? Suppose we had equalities instead of "less than" signs in the previous equations. Then, we would have a *linear* dynamical system: $\vec{p}(t) = \mathbf{S}\vec{p}(t-1) = \mathbf{S}^2\vec{p}(t-2) = \ldots = \mathbf{S}^t\vec{p}_0$. We could apply Lemma 4 to this system: $|\lambda_{1,\mathbf{S}}| < 1$ (from the statement of this Lemma) so the system would be stable and would converge to the fixed point $\vec{p}(t) = \vec{0}$ exponentially quickly. Thus, $\mathbf{S}^t\vec{p}_0 \to \vec{0}$ exponentially quickly.

Combining this with Equation 4.17, $\vec{0} \leq \vec{p}(t) < \mathbf{S}^t \vec{p}_0 \to \vec{0}$, and thus, $\vec{p}(t) \to 0$. So, $p_i(t) \to 0$ exponentially quickly, and from Lemma 5, this implies that $q_i(t) \to \frac{\gamma_i}{\gamma_i + \delta_i}$. Thus, we reach the fixed point $\left(p_i(t) = 0, q_i(t) = \frac{\gamma_i}{\gamma_i + \delta_i}\right)$ exponentially quickly, irrespective of the starting state. □

4.6 Summary

Under what conditions does information survive in a sensor or P2P network with node and link failures? As with the study of viral propagation, we could formulate the problem as a dynamical system, showing the generality of this framework. The dynamical system equations lead to a threshold condition below which a "datum" is expected to disappear from the network exponentially quickly, but above which it may survive for a long time. Several corollaries and special cases were used to demonstrate the intuitiveness of this result; in particular, we showed that the epidemic threshold for the SIS model of viral propagation is a special case of our information threshold result. Experiments on a variety of synthetic and real-world datasets, both sensor and P2P networks, show the accuracy of our results.

Chapter 5

Automatically grouping correlated nodes using Cross-Associations

"How can we automatically find natural node groups in a large graph?"

Clustering is one of the most common methods of data analysis, and it has many applications in graph mining. For example, given a list of customers and the products they buy, we might want to find customer "groups," and the product "groups" they are most interested in. This "segmenting" of the customer base can bring out the major customer "behaviors," and is useful in visualizing the data at a glance.

Clustering has applicability in a wide range of datasets. Apart from the customer-product example above, we can clusters data on:

- *Users versus their preferences:* With the growing importance of user-personalization on the Web, finding user groups and preference groups can be very useful.
- *Bank accounts versus transactions:* Individual bank accounts can be tagged according to the "transaction types" that they engage in. Deviation from this behavior can be used to trigger fraud detection systems.
- *Documents versus the words in them:* Here, we could find clusters of document groups (say, science fiction novels and thrillers), based on the word groups that occur most frequently in them. A user who prefers one document can then be recommended another document in that group.
- *Social networks:* A social network can describe relationships between individuals: who trusts whom, who meets whom, who has financial transactions with whom, and so on. Grouping people based on such information could be useful in, say, detection of money laundering rings in financial transaction networks.

Apart from the examples above, we can imagine many other instances where graph clustering

would be useful: in graph partitioning and community detection on the Web, in collaborative filtering applications, in microarray data analysis and so on.

Note that these examples include both bipartite graphs (such as users-vs-preferences) and self-graphs (such as social networks). In fact, any setting that has a *many-to-many* relationship in database terminology can be represented as a graph, so a graph clustering algorithm would be applicable on a wide variety of datasets. We focus only on *unweighted* graphs, which can be represented as adjacency matrices with 0/1 entries (see Chapter 2). Hence, from now on, we use the terms "graph" (with nodes) and "matrix" (with rows and columns) interchangeably.

A good graph clustering algorithm should have the following properties:

- **(P1)** It should automatically figure out the number of clusters.
- **(P2)** It should cluster the rows and columns simultaneously.
- **(P3)** It should be scalable.
- **(P4)** It should allow incremental updates.
- **(P5)** It should apply to both self-graphs and bipartite graphs. For bipartite graphs, we have row and column clusters, representing node groups in the "from" and "to" parts of the bipartite graph. For self-graphs, we might have an additional condition that the row and column clusters be identical (i.e., only "node" groups, instead of "from" and "to" groups).

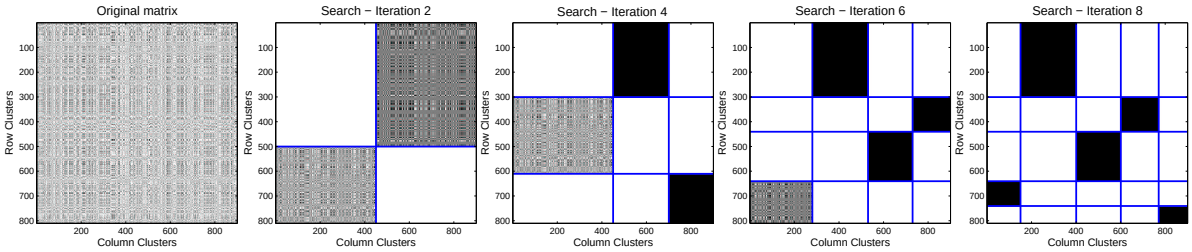
In this Chapter, we will describe an algorithm which has all of these properties. In addition, the same clustering algorithm can be easily extended to mine the data even further. Specifically, the goals of our algorithm are:

- **(G1)** Find clusters.
- **(G2)** Find outliers.
- **(G3)** Compute inter-cluster distances.

Intuitively, we seek to group rows and edges (i.e., nodes) so that the adjacency matrix is divided into rectangular/square regions as "similar" or "homogeneous" as possible. The homogeneity would imply that the graph nodes in that (row or column) group are all "close" to each other, and the density of each region would represent the strength of connections between groups. These regions of varying density, which we call *cross-associations*, would succinctly summarize the underlying structure of associations between nodes.

In short, our method will take as input a matrix like in Figure 5.1(a), and it will quickly and automatically (i) determine a good number of row groups k and column groups ℓ and (ii) re-order the rows and columns, to reveal the hidden structure of the matrix, like in Figure 5.1(e). Then, it will use the structure found in the previous step to automatically find abnormal ("outlier") edges, and also to compute the "distances" between pairs of groups.

We will first discuss related work in Section 5.1. Then, in Section 5.2, we formulate our data description model, and a corresponding *cost function* for each possible clustering. Based on this,



(a) Original matrix (b) Iteration 1 (c) Iteration 2 (d) Iteration 3 (e) Iteration 4

Figure 5.1: *Snapshots of algorithm execution:* Starting with the matrix in plot (a), the clustering is successively refined and the number of clusters increased till we reach the correct clustering in plot (e).

we describe our *automatic*, *parameter-free* algorithm for finding good clusters in Section 5.3. We then use these clusters to find outliers and compute inter-cluster distances in Section 5.4. Experimental results are provided in Section 5.5. Section 5.6 gives detailed proofs, followed by a summary in Section 5.7.

5.1 Related Work

There are numerous settings where we want to find patterns, correlations and rules, and time-tested tools exist for most of these tasks. However, with a few exceptions, all require tuning and human intervention, thus failing on property **(P1)**. We will discuss several of these approaches below.

Clustering: Traditional clustering approaches group along one dimension only: given a collection of n points in m dimensions, they find "groupings" of the n points. This setting makes sense in several domains (for example, if the m dimensions have an inherent ordering), but it is different from our problem setting.

Also, most of the algorithms require a user-given parameter, such as the number of clusters k in the popular k-means approach. The problem of finding k is a difficult one and has attracted attention recently; examples include X-means [Pelleg and Moore, 2000], which uses the BIC heuristic, and G-means [Hamerly and Elkan, 2003], which assumes a mixture of Gaussians (an often reasonable assumption, but which may not hold for binary matrices).

There are many other recent clustering algorithms too, including CURE [Guha et al., 1998], BIRCH [Zhang et al., 1996], Chameleon [Karypis et al., 1999], [Hinneburg and Keim, 1998], and some k-means variants such as k-harmonic means [Zhang et al., 2000] and spherical k-means [Dhillon and Modha, 2001]; see also [Han and Kamber, 2000].

However, they often need k as user input, or focus on clustering along one dimension only, or suffer from the dimensionality curse (like the ones that require a co-variance matrix); others may not scale up for large datasets. Also, in our case, the points and their corresponding vectors are semantically related (each node occurs as a point *and* as a component of each vector); most clustering methods do not consider this.

Thus, while these methods are applicable in many settings, they do not match our required properties.

Co-clustering: Recent work on Information-theoretic Co-clustering [Dhillon et al., 2003] (called ITCC from now on) is more closely related to ours; however, the differences are significant.

- ITCC focuses on contingency tables, not binary matrices.
- The main idea is based on lossy compression, whereas we employ a lossless compression scheme.
- ITCC seeks matrix approximations that minimize the KL-divergence to the original matrix, while we employ an MDL-based approach and a model that describes the data exactly.
- Finally, ITCC assumes knowledge of the number of clusters.

As of this writing, it is not clear how to apply MDL philosophy to the frameworks in [Dhillon et al., 2003, Friedman et al., 2001, Hofmann, 1999]; in contrast, our formulation is designed from grounds-up to be amenable to a parameter-free formulation.

More remotely related are matrix decompositions based on aspect models (one of the most prominent being PLSA [Hofmann, 1999], which again minimizes KL-divergence). These also assume that the number of "aspects" is given.

Graph partitioning: The prevailing methods are METIS [Karypis and Kumar, 1998b], and spectral partitioning [Andrew Y.Ñg, 2001]. Both approaches have attracted a lot of interest and attention; however, both need the user to specify k, that is, how many pieces should the graph be broken into. Some methods have recently been suggested to find k [Ben-Hur and Guyon, 2003, Tibshirani et al., 2001], but they need to sample from the data many times and then process/cluster those samples, which can become very slow and infeasible for large datasets. In addition, these graph partitioning techniques typically require a measure of imbalance between the two pieces of each split.

Several other partitioning techniques have also been suggested. The *Markov Clustering* [van Dongen, 2000] method uses random walks, but is slow. Girvan and Newman [Girvan and Newman, 2002] iteratively remove edges with the highest "stress" to eventually find disjoint communities, but the algorithm is again slow; Clauset et al. [Clauset et al., 2004] suggest a faster algorithm but the number of clusters must still be specified by the user. Flake et al. [Flake et al., 2000] use the max-flow min-cut formulation to find communities around a seed node; however, the selection of seed nodes is not fully automatic.

Market-basket analysis / frequent itemsets: These have been very important data mining areas, and have attracted a lot of research [Agrawal and Srikant, 1994, Han et al., 2004, Han and Kamber, 2000]. However, the user needs to specify the "support" parameter. The related work on "interestingness" [Tuzhilin and Adomavicius, 2002] still does not answer the question of "support."

Information retrieval and LSI: The pioneering method of LSI [Deerwester et al., 1990] used SVD on the term-document matrix. Again, the number k of eigenvectors/concepts to keep is up to the user ([Deerwester et al., 1990] empirically suggest about 200 concepts). Additional matrix

decompositions include the Semi-Discrete Decomposition, (SDD) [Kolda and O'Leary, 1998], PLSA [Hofmann, 1999], the clever use of random projections to accelerate SVD [Papadimitriou et al., 1998], and many more. However, they all fail on property **(P1)**.

Other domains: Also related to graphs in several settings is the work on conjunctive clustering [Mishra et al., 2003] requires density (i.e., "homogeneity") and overlap parameters, as well as community detection [Reddy and Kitsuregawa, 2001], among many others. Finally, there are several approaches to cluster micro-array data (e.g., [Tang and Zhang, 2003]).

In conclusion, the above methods miss one or more of our prerequisites, typically **(P1)**. Next, we present our solution.

5.2 Data Description Model

First, we will define some terminology (see Table 5.1). Let the graph to be clustered be G, with $\mathbf{A}$ being its adjacency matrix of size $M \times N$ ($M = N$ if G is a self-graph). Let us index the rows as $1, 2, \ldots, M$ and columns as $1, 2, \ldots, N$.

A cross-association (Φ, Ψ) of k row clusters and ℓ column clusters is defined by:

$$\begin{aligned} \Phi : \{1, 2, \ldots, M\} &\rightarrow \{1, 2, \ldots, k\} \\ \Psi : \{1, 2, \ldots, N\} &\rightarrow \{1, 2, \ldots, \ell\} \end{aligned} \tag{5.1}$$

where Φ and Ψ are the mappings of rows to row clusters and columns to column clusters. For a self-graph, where rows and columns represent identical nodes, we might want a single set of node clusters instead of separate row and column clusters (we will call this the IDENTICAL case); then, $\Phi = \Psi$ and $k = \ell$.

Given (Φ, Ψ), we can re-order $\mathbf{A}$ so that rows/columns from the same cluster are grouped together, creating $k \cdot \ell$ rectangular/square blocks (called cross-associates), which we denote by $\mathbf{A}_{i,j}$, with $i = 1 \ldots M$ and $j = 1 \ldots N$. Each block $\mathbf{A}_{i,j}$ has size $n(\mathbf{A}_{i,j}) = a_i \cdot b_j$, and contains $n_0(\mathbf{A}_{i,j})$ "zeros" and $n_1(\mathbf{A}_{i,j})$ "ones." From this, we can calculate the density $P_1(\mathbf{A}_{i,j})$ of "ones" in the block. High(low) densities imply that certain groups have stronger(weaker) connections with other groups.

Given an initial matrix $\mathbf{A}$ (as in Figure 5.1(a)), we want to automatically find clusters (as in Figure 5.1(e)). How can we estimate the "correct" number of row and column clusters, and the memberships of these clusters?

5.2.1 Main Idea

To compress the matrix, we would prefer to have only a few blocks, each of them being very homogeneous. However, having more clusters lets us create more homogeneous blocks (at the extreme, having 1 cluster per node gives $M \cdot N$ perfectly homogeneous blocks of size 1×1). Thus, the best compression scheme must achieve a tradeoff between these two factors, and this tradeoff point indicates the best values for k and ℓ.

Symbol	Description
G	Unweighted graph
$\mathbf{A}$	Binary adjacency matrix corresponding to G (square for self-graphs, possibly rectangular for bipartite graphs)
M, N	Number of rows and columns in $\mathbf{A}$ ($M = N$ for a self-graph)
k, ℓ	Number of row and column groups
E	Number of edges in G
k^*, ℓ^*	Optimal number of groups
(Φ, Ψ)	Cross-association
$\mathbf{A}_{i,j}$	Cross-associate (submatrix)
a_i, b_j	Dimensions of $\mathbf{A}_{i,j}$
$n(\mathbf{A}_{i,j})$	Number of elements $n(\mathbf{A}_{i,j}) := a_i b_j$
$n_0(\mathbf{A}_{i,j}), n_1(\mathbf{A}_{i,j})$	Number of 0, 1 elements in $\mathbf{A}_{i,j}$
$n_1(\mathbf{A})$	Number of 1 elements in $\mathbf{A}$; $n_1(\mathbf{A}) = E$
$P_0(\mathbf{A}_{i,j}), P_1(\mathbf{A}_{i,j})$	Densities of 0, 1 in $\mathbf{A}_{i,j}$
$P_{i,j}(t)$	$P_{i,j}(t) = P_1(\mathbf{A}_{i,j}, t)$ is the density at time t
$H(p)$	Binary Shannon entropy function
$C(\mathbf{A}_{i,j})$	Code cost for $\mathbf{A}_{i,j}$
$T(\mathbf{A}; k, \ell, \Phi, \Psi)$	Total cost for $\mathbf{A}$

Table 5.1: Table of symbols

We accomplish this by a novel application of the overall MDL philosophy, where the compression costs are based on the number of bits required to transmit both the "summary" of the row/column groups, as well as each block given the groups. Thus, *the user does not need to set any parameters*; our algorithm chooses them so as to minimize these costs. We discuss the exact cost function below.

5.2.2 The Cost Function

In conformance with the MDL philosophy, we design a lossless code for the data, and use the number of encoding bits as the cost function. This cost has two parts: (1) the *description cost* of describing the cross-association (Φ, Ψ) and the blocks formed by it, and (2) the *code cost* of describing the entire matrix *given* the cross-association.

Description cost: When we have separate row and column clusters, the description of the cross-association consists of the following parts:

1. Send the matrix dimensions m and n using $\log^\star(m) + \log^\star(n)$, where $\log^\star$ is the universal code for integers [Rissanen, 1983] defined by

$$\log^\star(x) = \log_2(x) + \log_2 \log_2(x) + \ldots \tag{5.2}$$

where only the positive terms are retained. This term is independent of the cross-association, and, hence, while useful for actual transmission of the data, will not figure in our cost function.

2. Send the row and column permutations using $M\lceil \log M \rceil$ and $N\lceil \log N \rceil$ bits, respectively. Again, this term is also independent of cross-association.

3. Send the number of groups: (k, ℓ) in $\log^\star k + \log^\star \ell$ bits.

4. Send the number of rows in each row group and also number of columns in each column group. Let us suppose that $a_1 \geq a_2 \geq \ldots \geq a_k \geq 1$ and $b_1 \geq b_2 \geq \ldots \geq b_\ell \geq 1$. Compute

$$\begin{aligned} \bar{a}_i &:= \left(\sum_{t=i}^{k} a_t \right) - k + i, \quad i = 1, \ldots, k-1 \\ \bar{b}_j &:= \left(\sum_{t=j}^{\ell} b_t \right) - \ell + j, \quad j = 1, \ldots, \ell - 1 \end{aligned}$$

   Now, the desired quantities can be sent using the following number of bits:

$$\sum_{i=1}^{k-1} \lceil \log \bar{a}_i \rceil + \sum_{j=1}^{\ell-1} \lceil \log \bar{b}_j \rceil$$

5. For each cross-associate $\mathbf{A}_{i,j}$, $i = 1, \ldots, k$ and $j = 1, \ldots, \ell$, send the number of "ones" $n_1(\mathbf{A}_{i,j})$ in it using $\lceil \log(a_i b_j + 1) \rceil$ bits.

Summing all of these:

$$\begin{aligned} \text{Description Cost} \;=\; & \log^\star k + \log^\star \ell \\ & + \sum_{i=1}^{k-1} \lceil \log \bar{a}_i \rceil + \sum_{j=1}^{\ell-1} \lceil \log \bar{b}_j \rceil \\ & + \sum_{i=1}^{k} \sum_{j=1}^{\ell} \lceil \log(a_i b_j + 1) \rceil \end{aligned}$$

In the IDENTICAL case, we must transmit only node clusters instead of row and column clusters, and the equation is modified to reflect this:

$$\begin{aligned} \text{Description Cost when } (\Phi = \Psi) \;=\; & \log^\star k \\ & + \sum_{i=1}^{k-1} \lceil \log \bar{a}_i \rceil \\ & + \sum_{i=1}^{k} \sum_{j=1}^{k} \lceil \log(a_i a_j + 1) \rceil \end{aligned}$$

Code cost: Suppose that the entire preamble specified above (containing information about the square and rectangular blocks) has been sent. We now transmit the actual matrix given this information. The number of bits $C(\mathbf{A}_{i,j})$ to encode each block depends only on its density $P_1(\mathbf{A}_{i,j})$:

$$\begin{aligned} C(\mathbf{A}_{i,j}) &= n(\mathbf{A}_{i,j}) \cdot H\left(P_1\left(\mathbf{A}_{i,j}\right)\right) \\ &= n_1\left(\mathbf{A}_{i,j}\right) \cdot \log \frac{1}{P_1\left(\mathbf{A}_{i,j}\right)} + n_0\left(\mathbf{A}_{i,j}\right) \cdot \log\left(1 - \frac{1}{P_1\left(\mathbf{A}_{i,j}\right)}\right) \end{aligned} \tag{5.3}$$

Summing over all blocks,

$$\text{Code cost} = \sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}) \tag{5.4}$$

For the IDENTICAL case, the equation is modified slightly:

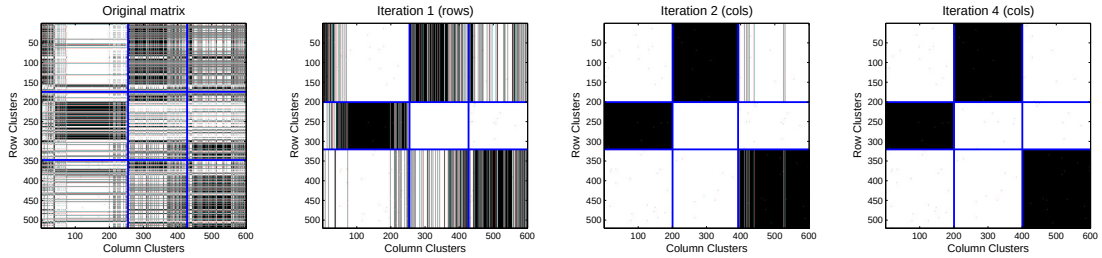
$$\text{Code cost when } (\Phi = \Psi) = \sum_{i=1}^{k}\sum_{j=1}^{k} C(\mathbf{A}_{i,j}) \tag{5.5}$$

Final cost function: Summing the description cost and the code cost gives us the total cost of encoding the matrix $\mathbf{A}$ using the cross-association (Φ, Ψ):

$$\begin{aligned} \text{Total cost } T(\mathbf{A}; k, \ell, \Phi, \Psi) &= \log^\star k + \log^\star \ell \\ &\quad + \sum_{i=1}^{k-1} \lceil \log \bar{a}_i \rceil + \sum_{j=1}^{\ell-1} \lceil \log \bar{b}_j \rceil \\ &\quad + \sum_{i=1}^{k}\sum_{j=1}^{\ell} \lceil \log(a_i b_j + 1) \rceil \\ &\quad + \sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}) \end{aligned} \tag{5.6}$$

For the IDENTICAL case:

$$\begin{aligned} \text{Total cost when } (\Phi = \Psi)\ T(\mathbf{A}; k, \Phi) &= \log^\star k \\ &\quad + \sum_{i=1}^{k-1} \lceil \log \bar{a}_i \rceil \\ &\quad + \sum_{i=1}^{k}\sum_{j=1}^{k} \lceil \log(a_i a_j + 1) \rceil \\ &\quad + \sum_{i=1}^{k}\sum_{j=1}^{k} C(\mathbf{A}_{i,j}) \end{aligned} \tag{5.7}$$



(a) Original groups (b) Step 2 (c) Step 4 (d) Step 4

Figure 5.2: *Iterations of the* SHUFFLE *algorithm:* Holding k and ℓ fixed ($k = \ell = 3$), we repeatedly apply Steps 2 and 4 of the SHUFFLE algorithm until no improvements are possible (Step 6). Iteration 3 (Step 2) is omitted, since it performs no swapping. To potentially decrease the cost further, we must increase k or ℓ or both, which is what the SPLIT algorithm does.

5.3 (G1) Graph Clustering Algorithm

The equations mentioned above assign a cost to each possible clustering; given two distinct clusterings, we can use the cost function to choose between the two. How can we use this to *find* a good clustering? In other words, how do we pick the best numbers of clusters k^* and ℓ^*, and the memberships of these clusters?

We propose an iterative two-step scheme to answer this question:

1. SHUFFLE (inner loop): For a given k and ℓ, find a good arrangement (i.e., cross-association).
2. SPLIT (outer loop): Efficiently search for the best k and ℓ ($k, \ell = 1, 2, \ldots$).

We present each in the following sections.

5.3.1 SHUFFLE (Inner Loop)

SHUFFLE (Figure 5.3) is a simple and efficient alternating minimization algorithm that yields a local minimum for the code cost (Equation 5.4, or 5.5 for the IDENTICAL case). In the regions where SHUFFLE is performed, the code cost typically dominates the description cost; thus, lowering the code cost also lowers the total cost. Figure 5.2 shows snapshots of SHUFFLE iterations on an example dataset.

Theorem 5. *After each iteration of* SHUFFLE*, the code cost either decreases or remains the same:* $\sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}(t)) \geq \sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}(t+1)) \geq \sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}(t+2))$.

Proof. The proof is provided in Section 5.6. □

The algorithm is analogous for the IDENTICAL case: instead of considering row splices x^j and

column splices y^i separately, we must consider them together:

$$\begin{aligned}\Phi_{t+1}(x) \;=\; & \arg\min_{1\le i\le k} \sum_{j=1}^{k} \Bigg[n_1(x^j) \log \frac{1}{P_{i,j}(t)} + n_0(x^j) \log \left(1 - \frac{1}{P_{i,j}(t)}\right) \\ & + n_1(y^j) \log \frac{1}{P_{j,i}(t+1)} + n_0(y^j) \log \left(1 - \frac{1}{P_{j,i}(t+1)}\right) \Bigg] \\ & + \mathbf{A}_{x,x} \left[\log P_{i,\Phi_t(x)}(t) + \log P_{\Phi_t(x),i}(t) - \log P_{i,i}(t) \right] \\ & + (1 - \mathbf{A}_{x,x}) \left[\log \left(1 - P_{i,\Phi_t(x)}(t)\right) + \log \left(1 - P_{\Phi_t(x),i}(t)\right) - \log \left(1 - P_{i,i}(t)\right) \right]\end{aligned}$$

where $\mathbf{A}_{x,x}$ is one if the node has a link to itself (a *self-loop*), and zero otherwise. Thus, the first two terms are the same as in Figure 5.3, and the last two terms take care of "double-counting" of the self-loop.

5.3.2 SPLIT (Outer Loop)

Every time SHUFFLE converges to a local minimum of the code cost, we can search for better values of k and ℓ. This search should attempt to lower the total cost (Equation 5.6 or 5.7), and we can employ any integer or even continuous optimization algorithm (gradient descent, Nelder-Mead, simulated annealing). We experimented with several alternatives, and obtained the best results with the SPLIT algorithm, shown in Figure 5.4. The IDENTICAL case is analogous. Figure 5.1 gives example snapshots from the execution of the full algorithm, with each plot showing the results after one iteration of SPLIT, followed by SHUFFLE iterations.

Theorem 6. *On each iteration of* SPLIT*, the code cost either decreases or remains the same: If* $\mathbf{A} = [\mathbf{A}_1 \mathbf{A}_2]$*, then* $C(\mathbf{A}_1) + C(\mathbf{A}_2) \le C(\mathbf{A})$.

Proof. The proof is provided in Section 5.6. □

5.3.3 Matching the desired properties

To summarize, our algorithm finds clusters in a graph by repeatedly searching for better values for k and ℓ (the numbers of groups), and then rearranging the adjacency matrix to decrease the cost, while keeping the numbers of groups fixed. We have thus matched goal **(G1)**, while matching all of our desired properties:

- **(P1)** By Theorems 5 and 6, each of these steps maintains or decreases the code cost. However, the description complexity evidently increases with k and ℓ. The total cost metric (Equation 5.6 or 5.7 depending on the problem setting) biases the algorithm towards fewer clusters, and provides an automatic stopping criterion. Thus, the algorithm requires no human intervention, and is *completely automatic*.
- **(P2)** By construction, the algorithm *treats rows and columns equally and simultaneously*.

Algorithm SHUFFLE (Finding better (Φ, Ψ) given k, ℓ):

1. Let t denote the iteration index. Initially, set $t = 0$. If no (Φ_0, Ψ_0) is provided, start with an arbitrary (Φ_0, Ψ_0) mapping nodes into k node groups. For this initial partition, compute the submatrices $\mathbf{A}_{i,j}(t)$, and the corresponding distributions $P_1(\mathbf{A}_{i,j}, t) \equiv P_{i,j}(t)$.

2. We will now hold column assignments, namely, Ψ_t, fixed. For every row x, splice it into ℓ parts each corresponding to one of the column groups. Denote them as $x^1, \ldots, x^\ell$. For each of these parts, compute $n_0(x^j)$ and $n_1(x^j)$, where $j = 1, \ldots, \ell$. Now, shuffle row x to row group $\Phi_{t+1}(x)$ such that:

$$\Phi_{t+1}(x) = \arg\min_{1 \le i \le k} \sum_{j=1}^{\ell} \left[n_1(x^j) \log \frac{1}{P_{i,j}(t)} + n_0(x^j) \log \left(1 - \frac{1}{P_{i,j}(t)} \right) \right] \tag{5.8}$$

3. With respect to cross-association (Φ_{t+1}, Ψ_t), recompute the matrices $\mathbf{A}_{i,j}(t+1)$, and corresponding distributions $P_1(\mathbf{A}_{i,j}, t+1)) \equiv P_{i,j}(t+1)$.

4. For this step, we will hold row assignments, namely, Φ_{t+1}, fixed. For every column y, splice it into k parts each corresponding to one of the column groups. Denote them as $y^1, \ldots, y^k$. For each of these parts, compute $n_0(y^i)$ and $n_1(y^i)$, where $i = 1, \ldots, k$. Now, assign y to column group $\Psi_{t+2}(y)$ such that:

$$\Psi_{t+2}(y) = \arg\min_{1 \le j \le \ell} \sum_{i=1}^{k} \left[n_1(y^i) \log \frac{1}{P_{i,j}(t+1)} + n_0(y^i) \log \left(1 - \frac{1}{P_{i,j}(t+1)} \right) \right] \tag{5.9}$$

5. For the new cross-association (Φ_{t+1}, Ψ_{t+2}), recompute the matrices $\mathbf{A}_{i,j}(t+2)$, and corresponding distributions $P_1(\mathbf{A}_{i,j}, t+2)) \equiv P_{i,j}(t+2)$.

6. If there is no decrease in total cost, stop; otherwise, set $t = t + 2$, go to step 2, and iterate.

Figure 5.3: Algorithm SHUFFLE (inner loop)

- **(P3)** The overall complexity is $O\left(E \cdot (k^* + \ell^*)^2\right)$, if we ignore the number of SHUFFLE iterations I (in practice, $I \le 20$ is always sufficient). Thus, it is linear in the number of edges, and hence *scalable*.

- **(P4)** When new nodes are obtained (such as from new crawls for a Web graph), we can put them into the clusters which minimize the increase in total encoding cost. Similarly, when new edges are found, the corresponding nodes can be reassigned to new clusters. The algorithm can then be run again with this initialization till it converges. Similar methods apply for node/edge deletions. Thus, *new additions or deletions can be handled incrementally*.

- **(P5)** *The algorithm is applicable to both self-graphs and bipartite graphs*. When we require a single set of node clusters instead of separate row and column clusters, the extra constraint can be handled easily, as we demonstrated for the IDENTICAL case.

Algorithm SPLIT (Choosing better k and ℓ):

1. Let T denote the search iteration index. Start with $T = 0$ and $k^0 = \ell^0 = 1$.

2. At iteration T, increase the number of row groups: $k(T+1) = k(T)+1$. Split the row group r with maximum entropy, i.e.,

$$r := \arg\max_{1 \le i \le k} \sum_{1 \le j \le \ell} n_1\left(\mathbf{A}_{i,j}\right) \cdot H\left(P_{i,j}\right)$$

   Construct an initial label map Φ_{T+1} as follows: For every row x that belongs to row group r (i.e., $\Phi_T(x) = r$), place it into the new group $k(T+1)$ (i.e., set $\Phi_{T+1}(x) = k(T+1)$) if and only if it decreases the per-row entropy of the group r, i.e., if and only if

$$\sum_{1 \le j \le \ell} \frac{n_1\left(\mathbf{A}'_{i,j}\right) \cdot H\left(P'_{r,j}\right)}{a_r - 1} < \sum_{1 \le j \le \ell} \frac{n_1\left(\mathbf{A}_{i,j}\right) \cdot H\left(P_{r,j}\right)}{a_r}$$

   where $\mathbf{A}'_{r,j}$ is $\mathbf{A}_{r,j}$ without row x. Otherwise let $\Phi_{T+1}(x) = r = \Phi_T(x)$. If we move the row to the new group, we also update $D_{r,j}$ (for all $1 \le j \le \ell$).

3. Run SHUFFLE (the inner loop) with initial configuration (Φ_{T+1}, Ψ_T) to find a new cross-association and the corresponding total cost.

4. If there is no decrease in total cost, stop and return $(k^*, \ell^*) = (k^T, \ell^T)$—with corresponding cross-associations (Φ_T, Ψ_T). Otherwise, set $T = T + 1$ and continue.

5–7. Similar to steps 2–4, but with columns instead.

Figure 5.4: Algorithm SPLIT (outer loop)

5.3.4 Relationship with hierarchical clustering

Our algorithm finds one flat set of node groups. Even though each new group is formed by a split of a pre-existing group, the SHUFFLE iterations mix up the group memberships, and any hierarchy is destroyed. However, we could obtain a hierarchical clustering by (1) running the full algorithm to obtain a flat clustering, (2) separately applying the entire algorithm on each of these clusters, and (3) repeating. This is an avenue for future work.

5.4 Finding outlier edges and inter-cluster distances

Having found the underlying structure of a graph in the form of node groups (goal **(G1)**), we can utilize this information to further mine the data. Specifically, we want to detect outlier edges (goal **(G2)**) and compute inter-group "distances" (goal **(G3)**). Again, we use our information-theoretic approach to solve all these problems efficiently.

5.4.1 (G2) Outlier edges

Which edges between nodes are abnormal/suspicious? Intuitively, an outlier shows some deviation from normality, and so it should hurt attempts to compress data. Thus, an edge whose removal significantly reduces the total encoding cost is an outlier. Our algorithm is: find the block where removal of an edge leads to the maximum immediate reduction in cost (that is, no iterations of SHUFFLE and SPLIT are performed). All edges within that block contribute equally to the cost, and so all of them are considered outliers.

$$\text{“Outlierness” of edge } (u, v) := T(\mathbf{A}'; k, \ell, \Phi, \Psi) - T(\mathbf{A}; k, \ell, \Phi, \Psi) \quad (5.10)$$

where $\mathbf{A}'$ is $\mathbf{A}$ without the edge (u, v). This can be used to *rank* the edges in terms of their “outlierness”.

5.4.2 (G3) Computing inter-group “distances”

How “close” are two node groups to each other? Following our information theory footing, we propose the following criterion: If two groups are “close”, then combining the two into one group should not lead to a big increase in encoding cost. Based on this, we define “distance” between two groups as the relative increase in encoding cost if the two were merged into one:

$$Dist(i, j) := \frac{Cost(merged) - Cost(i) - Cost(j)}{Cost(i) + Cost(j)} \quad (5.11)$$

where only the nodes in groups i and j are used in computing costs. We experimented with other measures (such as the absolute increase in cost) but Equation 5.11 gave the best results.

To computing outliers and distances between groups, only the statistics of the final clustering need to be used (i.e., the block sizes $n_{i,j}$ and their densities $P_{i,j}$). Thus, both can be performed efficiently for large graphs.

5.5 Experiments

We did experiments to answer two key questions:

(Q1) How good is the quality of the clusters?

(Q2) How well do our algorithms find outlier edges?

(Q3) How well do our measures of inter-cluster “distances” work?

(Q4) Is our method scalable?

We carried out experiments on a variety of datasets, both synthetic and real-world (Table 5.2). The synthetic datasets were:

- *CAVE*: This represents a social network of "cavemen" [Watts, 1999], that is, a block-diagonal matrix of variable-size blocks (or "caves").
- *CUSTPROD*: This represents groups of customers and their buying preferences[1]
- *CAVE-NOISY*: A *CAVE* dataset with "salt-and-pepper" noise (10% of the number of edges).
- *NOISE*: This contains pure white noise.

All of these datasets were scrambled before being fed into the cross-associations program as input.

The real-world datasets were:

- *CLASSIC*: A bipartite graph of Usenet documents from Cornell's SMART collection, and the words present in them (see [Dhillon et al., 2003]). The documents belong to three distinct groups: MEDLINE (medicine), CISI (information retrieval), and CRANFIELD (aerodynamics).
- *GRANTS*: A set of NSF proposal documents from several disciplines (physics, bio-informatics, etc.), versus the words in their abstracts.
- *EPINIONS*: A "who-trusts-whom" social graph of `www.epinions.com` users [Domingos and Richardson, 2001].
- *CLICKSTREAM*: A graph of users and the URLs they clicked on [Montgomery and Faloutsos, 2001].
- *OREGON*: A graph of network connections between Autonomous Systems (AS), obtained from `http://topology.eecs.umich.edu/data.html`.
- *DBLP*: A co-citation and co-authorship graph extracted from `www.informatik.uni-trier.de/~ley/db`. The nodes are authors in the `SIGMOD, ICDE, VLDB, PODS` or `ICDT` (database conferences); two nodes are linked by an edge if the two authors have co-authored a paper or one has cited a paper by the other (thus, this graph is undirected).

Our implementation was done in MATLAB (version 6.5 on Linux) using sparse matrices. The experiments were performed on an Intel Xeon 2.8GHz machine with 1GB RAM.

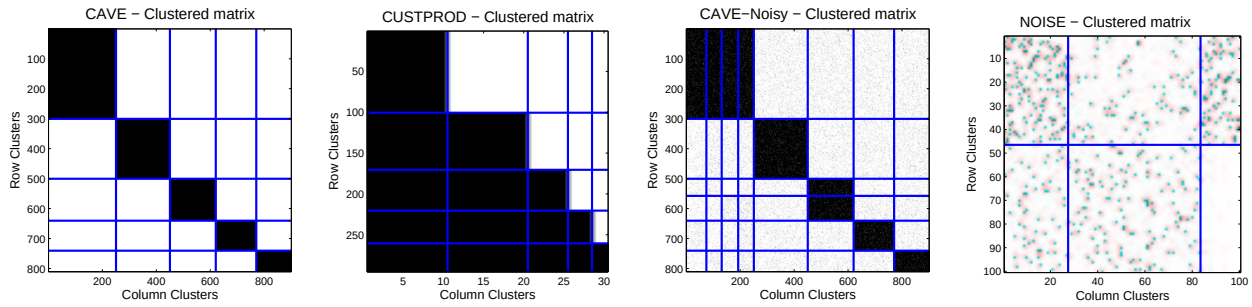
5.5.1 (Q1) Quality of clustering

Synthetic Datasets: Figure 5.5 shows the resulting cross-associations on the synthetic datasets. We can make the following observations:

[1]We try to capture market segments with heavily overlapping product preferences, like, say, "single persons," buying beer and chips, "couples," buying the above, plus frozen dinners, "families," buying all the above plus milk, and so on.

Dataset	Dimensions	Edges (directed)	Graph type
CAVE	810×900	162,000	Bipartite
CAVE-NOISY	810×900	171,741	Bipartite
CUSTPROD	295×30	5,820	Bipartite
NOISE	100×100	952	Self
CLASSIC	3,893×4,303	176,347	Bipartite
GRANTS	13,297×5,298	805,063	Bipartite
EPINIONS	75,888×75,888	508,960	Self
CLICKSTREAM	23,396×199,308	952,580	Bipartite
OREGON	11,461×11,461	65,460	Self
DBLP	6,090×6,090	175,494	Self

Table 5.2: Dataset characteristics.



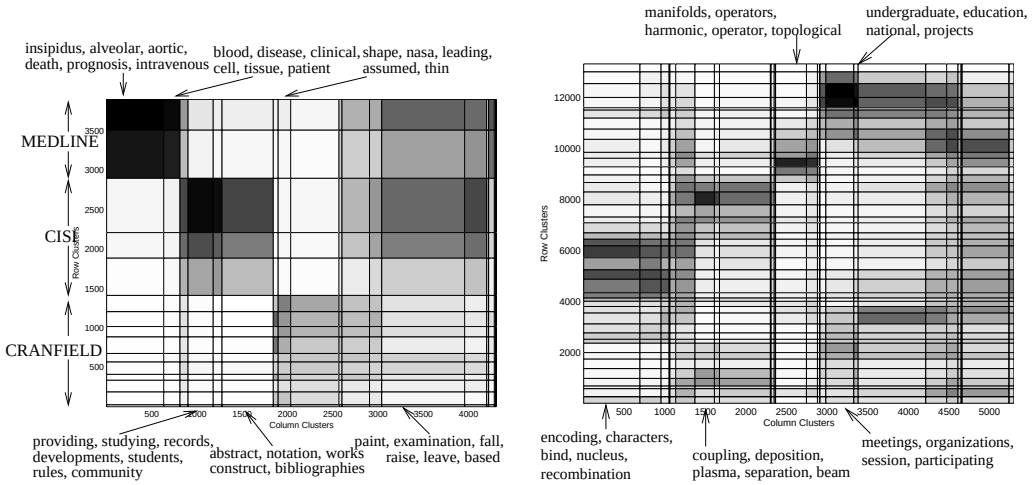
(a) *CAVE* (b) *CUSTPROD* (c) *CAVE-NOISY* (d) *NOISE*

Figure 5.5: *Cross-associations on synthetic datasets*: Our method gives the intuitively correct cross-associations for (a) *CAVE* and (b) *CUSTPROD*. Some extra groups are found for (c) *CAVE-NOISY*, which are explained by the patterns emerging due to randomness, such as the "almost-empty" and "more-dense" cross-associations for *NOISE* (d).

- *Exact results on noise-free datasets:* For *CAVE* and *CUSTPROD*, we get exactly the intuitively correct groups. This serves as a sanity check.
- *Robust performance on noisy datasets:* For *CAVE-NOISY*, we find some extra groups which, on closer examination, are picking up patterns in the noise. This is expected: it is well known that spurious patterns emerge, even when we have pure noise. Figure 5.5(d) confirms it: even in the *NOISE* matrix, our algorithm finds blocks of clearly lower or higher density. However, even with such significant noise, very few "spurious" groups are found.

CLASSIC*:* Figure 5.6(a) shows the clusters found in the *CLASSIC* dataset.

- *The results match the known groupings:* We see that the cross-associates are in agreement with the known document classes (left axis annotations). We also annotated some of the column groups with their most frequent words. Cross-associates belonging to the same document (row) group clearly follow similar patterns with respect to the word (column) groups.



(a) *CLASSIC* ($k^* = 15, \ell^* = 19$) (b) *GRANTS* ($k^* = 41, \ell^* = 28$)

Figure 5.6: *Cross-associations for CLASSIC and GRANTS*: Due to the dataset sizes, we show the Cross-associations via shading; darker shades correspond denser blocks (more ones). We also show the most frequently occurring words for several of the word (column) groups; these clearly belong to different categories of words.

For example, the MEDLINE row groups are most strongly related to the first and second column groups, both of which are related to medicine. ("insipidus," "alveolar," "prognosis" in the first column group; "blood," "disease," "cell," etc, in the second).

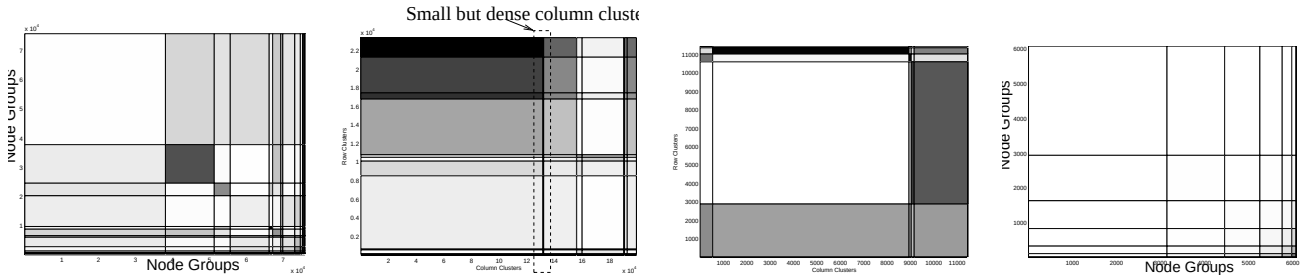
- *Previously unknown structure is revealed:* Besides being in agreement with the known document classes, the cross-associates *reveal further structure*. For example, the first word group consists of more "technical" medical terms, while second group consists of "everyday" terms, or terms that are used in medicine often, but not exclusively[2]. Thus, the second word group is more likely to show up in other document groups (and indeed it does, although not immediately apparent in the figure), which is why our algorithm separates the two.

GRANTS*:* Again, the results mirror those for the *CLASSIC* dataset:

- *Meaningful clusters are extracted:* Figure 5.6(b) shows the most common terms in several of the column clusters. They show that the groups found make intuitive sense: we detect clusters related to biology ("encoding," "recombination," etc), to physics ("coupling," "deposition", "plasma," etc), to material sciences, and to several other well-known topics.

EPINIONS, CLICKSTREAM, OREGON *and* DBLP*:* Figure 5.7 shows our results on all the other datasets. *EPINIONS* and *DBLP* are clustered under the IDENTICAL setting, so that the row and column clusters are the same. The results make intuitive sense:

[2]This observation is also true for nearly all of the (approximately) 600 and 100 words belonging to each group, not only the most frequent ones shown here.



(a) *EPINIONS* ($k^* = 19$) (b) *CLICKSTREAM* ($k^* = 15, \ell^* = 13$) (c) *OREGON* ($k^* = 9, \ell^* = 8$) (d) *DBLP* ($k^* = 8$)

Figure 5.7: *Cross-associations for other real-world datasets: CLICKSTREAM* and *OREGON* are clustered normally, while *EPINIONS* and *DBLP* were clustered under the IDENTICAL setting (i.e., identical row and column clusters). All the graphs get separated into two kinds of groups: large but very sparse, and small but very dense. The small dense clusters are often worthy of further analysis; for example, most well-known database researchers show up in the dense regions of plot (d).

- For example, for the *DBLP* dataset, the smallest group consists only of Michael Stonebraker, David DeWitt and Michael Carey; these are well-known people who have a lot of papers and citations. The other groups show decreasing number of connections but increasing sizes.
- Similarly, for the `EPINIONS` graph, we find a small dense "core" group which has very high connectivity, and then larger and less heavily-connected groupings.

5.5.2 (Q2) Outlier edges

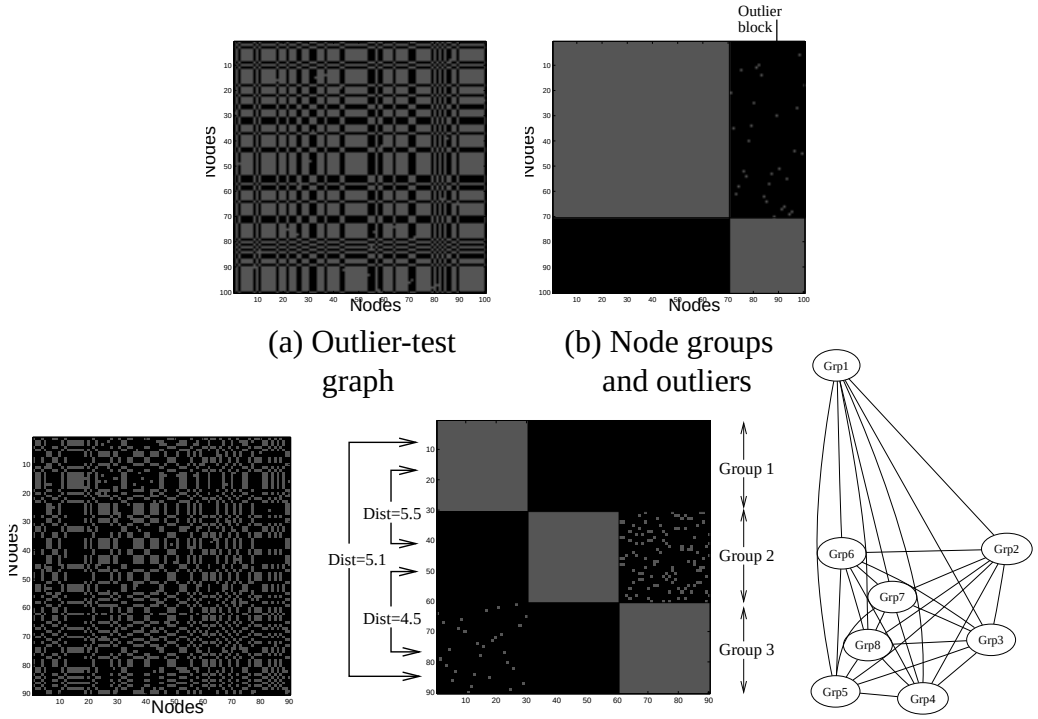
To test our algorithm for picking outliers, we use a synthetic dataset as in Figure 5.8(a). The node groups found are shown in 5.8(b). Our algorithm tags all edges whose removal would best compress the graph as outliers. Thus, all edges "across" the two groups are chosen as outliers under this principle (since all edges in a block contribute equally to the encoding cost), as shown in Figure 5.8(b). Thus, the intuitively correct outliers are found.

5.5.3 (Q3) Inter-cluster "distances"

To test for node-group distances, we use the graph in 5.8(c) with 5.8(d) showing the structure found. The three caves have equal sizes but the number of "bridge" edges between groups varies. This is correctly picked up by our algorithm, which ranks groups with more "bridges" as being closer to each other. Thus, groups 2 and 3 are tagged as the "closest" groups, while groups 1 and 2 are "farthest".

Figure 5.8(e) shows the inter-group distances between the clusters found for *DBLP*. The distances were computed using our algorithm and plotted using *Graphviz*[3]: longer lines imply larger

[3] `http://www.research.att.com/sw/tools/graphviz/`



(a) Outlier-test graph

(b) Node groups and outliers

(c) Distance-test graph

(d) Node groups and distances

(e) *DBLP* distances

Figure 5.8: *Outliers and group distances:* Plot (b) shows the node groups found for graph (a). Edges in the top-right block are correctly tagged as outliers. Plot (d) shows the node groups and group distances for graph (c). Groups 2 and 3 (having the most "bridges") are tagged as the closest groups. Similarly, groups 1 and 2 are the farthest, because they have no "bridges". Plot (e) is a visualization of the distances between groups for the *DBLP* dataset.

distances. The smallest group (Stonebraker, DeWitt and Carey) is "Grp8," and we that it occupies a central position exactly because it has many "bridges" with people from other groups. Similarly, "Grp1" is seen to be far from all other groups: this is because authors in "Grp1" have very few papers in the conferences included in this dataset, and thus their connections to the rest of the graph are very sparse.

5.5.4 (Q4) Scalability

We have seen that our algorithms give accurate and intuitive results on a variety of synthetic and real-world datasets. Here, we will verify their scalability.

Figure 5.9 shows results on a "caveman" graph with three caves. We see that the wall-clock time scales linearly with the number of edges E. Thus, our algorithms are scalable to large graph datasets.

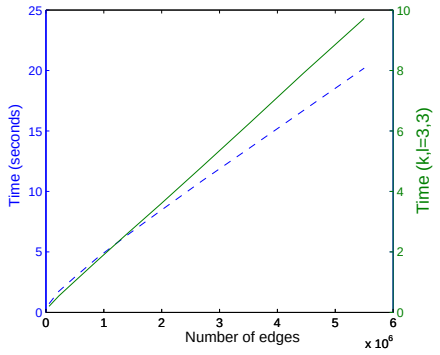


Figure 5.9: *Scalability:* We show timing results on a "caveman" graph with 3 caves. The plot shows wall-clock time vs. the number of edges E in the graph, for both SPLIT (dashed), as well as SHUFFLE for a particular (k, ℓ) (solid). We see that both are linear in E.

5.6 Details of Proofs

Theorem 2. *After each iteration of* SHUFFLE*, the code cost either decreases or remains the same:* $\sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}(t)) \geq \sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}(t+1)) \geq \sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}(t+2))$.

Proof. We shall only prove the first inequality, the second inequality will follow by symmetry between rows and columns. We will need some notation:

$$\begin{aligned} P_{i,j}(u,t) &= \begin{cases} P_{i,j}(t) & \text{if } u=1 \\ 1-P_{i,j}(t) & \text{if } u=0 \end{cases} \\ n_u(.) &= \begin{cases} n_1(.) & \text{if } u=1 \\ n_0(.) & \text{if } u=0 \end{cases} \end{aligned}$$

Now, we have:

$$\begin{aligned} \sum_{i=1}^{k}\sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}(t)) &= \sum_{i=1}^{k}\sum_{j=1}^{\ell}\Bigg[n_1\left(\mathbf{A}_{i,j}(t)\right)\cdot\log\frac{1}{P_1\left(\mathbf{A}_{i,j}(t)\right)} \\ &\qquad + n_0\left(\mathbf{A}_{i,j}(t)\right)\cdot\log\left(1-\frac{1}{P_1\left(\mathbf{A}_{i,j}(t)\right)}\right)\Bigg] \quad \text{(using Eq 5.3)} \\ &= \sum_{i=1}^{k}\sum_{j=1}^{\ell}\sum_{u=0}^{1} n_u(\mathbf{A}_{i,j}(t))\log\frac{1}{P_{i,j}(u,t)} \\ &= \sum_{i=1}^{k}\sum_{j=1}^{\ell}\sum_{u=0}^{1}\left[\sum_{x:\Phi_t(x)=i} n_u(x^j)\right]\log\frac{1}{P_{i,j}(u,t)} \\ &= \sum_{i=1}^{k}\sum_{x:\Phi_t(x)=i}\left[\sum_{j=1}^{\ell}\sum_{u=0}^{1} n_u(x^j)\log\frac{1}{P_{i,j}(u,t)}\right] \end{aligned}$$

$$
\begin{aligned}
&\overset{(a)}{\geq} \sum_{i=1}^{k} \sum_{x:\Phi_t(x)=i} \left[\sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(x^j) \log \frac{1}{P_{\Phi_{t+1}(x),j}(u,t)} \right] \\
&\overset{(b)}{=} \sum_{i=1}^{k} \sum_{x:\Phi_{t+1}(x)=i} \left[\sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(x^j) \log \frac{1}{P_{\Phi_{t+1}(x),j}(u,t)} \right] \\
&= \sum_{i=1}^{k} \sum_{j=1}^{\ell} \sum_{u=0}^{1} \left[\sum_{x:\Phi_{t+1}(x)=i} n_u(x^j) \right] \log \frac{1}{P_{i,j}(u,t)} \\
&= \sum_{i=1}^{k} \sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(\mathbf{A}_{i,j}(t+1)) \log \frac{1}{P_{i,j}(u,t)} \\
&\overset{(c)}{\geq} \sum_{i=1}^{k} \sum_{j=1}^{\ell} \sum_{u=0}^{1} n_u(\mathbf{A}_{i,j}(t+1)) \log \frac{1}{P_{i,j}(u,t+1)} \\
&= \sum_{i=1}^{k} \sum_{j=1}^{\ell} C(\mathbf{A}_{i,j}(t+1))
\end{aligned}
$$

where (a) follows from Step 2 of the Cross-association Algorithm; (b) follows by re-writing the outer two sums–since i is not used anywhere inside the $[\cdots]$ terms; and (c) follows from the non-negativity of the Kullback-Leibler distance. □

Theorem 3. *On each iteration of* SPLIT*, the code cost either decreases or remains the same: If* $\mathbf{A} = [\mathbf{A}_1 \mathbf{A}_2]$*, then* $C(\mathbf{A}_1) + C(\mathbf{A}_2) \leq C(\mathbf{A})$.

Proof. We have

$$
\begin{aligned}
C(\mathbf{A}) &= n(\mathbf{A}) H\left(P_{\mathbf{A}}\right) = n(\mathbf{A}) H\left(\frac{n_1(\mathbf{A})}{n(\mathbf{A})}\right) \\
&= H\left(\frac{P_{\mathbf{A}_1} n(\mathbf{A}_1) + P_{\mathbf{A}_2} n(\mathbf{A}_2)}{n(\mathbf{A})}\right) \\
&\geq n(\mathbf{A}) \left(\frac{n(\mathbf{A}_1)}{n(\mathbf{A})} H\left(P_{\mathbf{A}_1}\right) + \frac{n(\mathbf{A}_2)}{n(\mathbf{A})} H\left(P_{\mathbf{A}_2}\right)\right) \\
&= n(\mathbf{A}_1) H\left(P_{\mathbf{A}_1}\right) + n(\mathbf{A}_2) H\left(P_{\mathbf{A}_1}\right) \\
&= C(\mathbf{A}_1) + C(\mathbf{A}_2)
\end{aligned}
$$

where the inequality follows from the concavity of $H(\cdot)$ and the fact that $n(\mathbf{A}_1) + n(\mathbf{A}_2) = n(\mathbf{A})$ or $n(\mathbf{A}_1)/n(\mathbf{A}) + n(\mathbf{A}_2)/n(\mathbf{A}) = 1$. □

5.7 Summary

Our aims were very broad: we wanted to find underlying structure in a graph. To this end, we introduced a novel approach and proposed a general, intuitive model founded on lossless compres-

sion and information-theoretic principles. Based on this model, we provided algorithms for mining large graph datasets in a variety of ways. Specifically:

- We proposed one of the few methods for clustering nodes in a graph, that needs *no "magic" numbers*. Our algorithms figure out both the *number* of clusters in the data, and their *memberships*.
- Besides being fully automatic, our approach satisfies all of the desired properties **(P1)-(P5)**. This includes *scalability* to large datasets, ability to function *online* (with incremental data input), and applicability to *both self-graphs and bipartite graphs*.
- In addition to clustering, we also proposed efficient methods to *detect outlier edges* and to *compute inter-cluster "distances."* Both of these are important standalone data-mining problems in their own right.

Thus, we were able to achieve all our goals **(G1)-(G3)**. Experiments on a large set of synthetic and real-world datasets show the accuracy and efficiency of our methods; not only were we able to find known patterns in the data, but we could also detect *unknown* structure, as in the *CLASSIC* dataset.

Chapter 6

The R-MAT graph generator

"How can we quickly generate a synthetic yet realistic graph? How can we spot fake graphs and outliers?"

While we answered application-specific questions in the previous chapters, the focus of this chapter is on real-world graphs in general. What do real graphs look like? What patterns or "laws" do they obey? This is intimately linked to the problem of designing a good graph generator: a realistic generator is one which matches exactly these graph "laws." These patterns and generators are important for many applications:

- *Detection of abnormal subgraphs/edges/nodes:* Abnormalities should deviate from the "normal" patterns, so understanding the patterns of naturally occurring graphs is a prerequisite for detection of such outliers.
- *Simulation studies:* Algorithms meant for large real-world graphs can be tested on synthetic graphs which "look like" the original graphs. This is particularly useful if collecting the real data is hard or costly.
- *Realism of samples:* Most graph algorithms are super-linear on the node count, and thus prohibitive for large graphs. We might want to build a small sample graph that is "similar" to a given large graph. In other words, this smaller graph needs to match the "patterns" of the large graph to be realistic.
- *Extrapolation of data:* Given a real, evolving graph, we expect it to have x% more nodes next year; how will it then look like, assuming that our "laws" are still obeyed? For example, in order to test the next-generation Internet protocol, we would like to simulate it on a graph that is "similar" to what the Internet will look like a few years into the future.
- *Graph compression:* Graph patterns represent regularities in the data. Such regularities can be used to better compress the data.

Thus, we need to detect patterns in graphs, and then generate synthetic graphs matching such patterns automatically.

There are several desiderata from a graph generator:

(P1) *Realism*: It should only generate graphs that obey all (or at least several) of the above "laws", and it would match the properties of real graphs (degree exponents, diameters etc., that we shall discuss later) with the appropriate values of its parameters.

(P2) *Procedural generation*: Instead of creating graphs to specifically match some patterns, the generator should offer some *process* of graph generation, which automatically leads to the said patterns. This is necessary to gain insight into the process of graph generation in the real world: if a process cannot not generate synthetic graphs with the required patterns, it is probably not the underlying process (or at least the sole process) for graph creation in the real world.

(P3) *Parsimony*: It should have a few only parameters.

(P4) *Fast parameter-fitting*: Given any real-world graph, the model parameters should easily tunable by some published algorithms to generate graph similar to the input graph.

(P5) *Generation speed*: It should generate the graphs quickly, ideally, linearly on the number of nodes and edges.

(P6) *Extensibility*: The same method should be able to generate directed, undirected, and bipartite graphs, both weighted or unweighted.

This is exactly the main part of this work. We propose the *Recursive Matrix* (R-MAT) model, which naturally generates power-law (or "DGX" [Bi et al., 2001]) degree distributions. We show that it naturally leads to small-world graphs and also matches several other common graph patterns; it is recursive (=self-similar), and it has only a small number of parameters.

The rest of this chapter is organized as follows: Section 6.1 surveys the existing graph laws and generators. Section 6.2 presents the idea behind our R-MAT method. We discuss the properties of R-MAT graphs, and algorithms for graph generation and parameter-fitting under R-MAT. Section 6.3 gives the experimental results, where we show that R-MAT successfully mimics large real graphs. Section 6.4 provides details of proofs. A summary is provided in Section 6.5.

6.1 Related Work

The twin problems of finding graph patterns and building graph generators have attracted a lot of recent interest, and a large body of work has been done on both, not only by computer scientists, but also physicists, mathematicians, sociologists, and others. However, there is little interaction among these fields, with the result that they often use different terminology and do not benefit from each other's advances. In this section, we attempt to give a brief overview of the main ideas, with a focus on combining sources from all the different fields, to gain a coherent picture of the current state-of-the-art. The interested reader is also referred to some excellent and entertaining books on the topic [Barabási, 2002, Watts, 2003].

6.1.1 Graph Patterns and "Laws"

While there are many differences between graphs, some patterns show up regularly. Work has focused on finding several such patterns, which *together* characterize naturally occurring graphs. The main ones appear to be:

- Power laws (already seen in Chapter 2),
- Small diameters, and
- Community effects.

Power laws:

While the Gaussian distribution is common in nature, there are many cases where the probability of events far to the right of the mean is significantly higher than in Gaussians. In the Internet, for example, most routers have a very low degree (perhaps "home" routers), while a few routers have extremely high degree (perhaps the "core" routers of the Internet backbone) [Faloutsos et al., 1999]. Power-law distributions attempt to model this. Some of the following were defined in Chapter 2; we repeat the definitions here for completeness.

Definition 16 (Power Law). *Two variables x and y are related by a power law when their scatter plot is linear on a log-log scale:*

$$y(x) = c \cdot x^{-\gamma} \tag{6.1}$$

where c and γ are positive constants. The constant γ is often called the power law exponent.

Definition 17 (Degree Distribution). *The degree distribution of an undirected graph is a plot of the count c_k of nodes with degree k, versus the degree k, typically on a log-log scale. Occasionally, the fraction $\frac{c_k}{N}$ is used instead of c_k; however, this merely translates the log-log plot downwards. For directed graphs, outdegree and indegree distributions are defined similarly.*

Definition 18 (Scree Plot). *This is a plot of the eigenvalues (or singular values) of the adjacency matrix of the graph, versus their rank, using a log-log scale.*

Both degree distributions and scree plots of many real-world graphs have been found to obey power laws. Examples include the Internet AS and router graphs [Faloutsos et al., 1999, Govindan and Tangmunarunkit, 2000], the World-wide Web [Barabási and Albert, 1999, Kumar et al., 1999, Broder et al., 2000, Kleinberg et al., 1999], citation graphs [Redner, 1998], online social networks [Chakrabarti et al., 2004], and many others. Power laws also show up in the distribution of "bipartite cores" ($\approx$ communities) and the distribution of PageRank values [Brin and Page, 1998, Pandurangan et al., 2002]. Indeed, power laws appear to be a defining characteristic of almost all large real-world graphs.

Significance of power laws: The significance of power law distributions lies in the fact that they are *heavy-tailed*, meaning that they decay more slowly than exponential or Gaussian distributions. Thus, a power law degree distribution would be much more likely to have nodes with a very high degree (much larger than the mean) than the other two distributions.

Deviations from power laws: Pennock et al. [Pennock et al., 2002] and others have observed deviations from a pure power law distribution in several datasets. Two of the more common deviations are exponential cutoffs and lognormals. The exponential cutoff models distributions which look like power laws over the lower range of values on the x-axis, but decay exponentially quickly for higher values; examples include the network of airports [Amaral et al., 2000]. Lognormal distributions look like truncated parabolas on log-log scales, and model situations where the plot "dips" downwards in the lower range of values on the x-axis; examples include degree distributions of subsets of the WWW, and many others [Pennock et al., 2002, Bi et al., 2001].

Small diameters:

Several definitions of the term "graph diameter" exist: the definition we use is called the "effective diameter" or "eccentricity," and is closely related to the "hop-plot" of a graph; both of these are defined below. The advantages are twofold: (a) the "hop-plot" and "effective diameter" can be computed in linear time and space using a randomized algorithm [Palmer et al., 2002], and (b) this particular definition is robust in the presence of outliers.

Definition 19 (Hop-plot). *Starting from a node u in the graph, we find the number of nodes $N_h(u)$ in a neighborhood of h hops. We repeat this starting from each node in the graph, and sum the results to find the total neighborhood size N_h for h hops ($N_h = \sum_u N_h(u)$). The hop-plot is just the plot of N_h versus h.*

Definition 20 (Effective diameter). *This is the minimum number of hops in which some fraction (say, 90%) of all connected pairs of nodes can reach each other [Tauro et al., 2001].*

Significance of graph diameter: The diameters of many real-world graphs are very small compared to the graph size [Albert and Barabási, 2002]: only around 4 for the Internet AS-level graph, 12 for the Internet Router-level graph, 16 for the WWW, and the famous "six degrees of separation" in the social network. Any realistic graph generator needs to match this criterion.

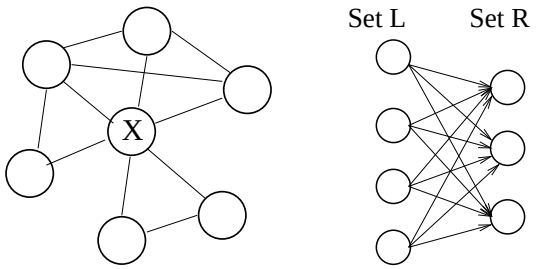
Community Effects:

Informally, a community is a set of nodes where each node is "closer" to the other nodes within the community than to nodes outside it. This effect has been found (or is believed to exist) in many real-world graphs, especially social networks [Moody, 2001, Schwartz and Wood, 1993].

Community effects have typically been studied in two contexts: (a) local one-hop neighborhoods, as characterized by the *clustering coefficient*, and (b) node groups with possibly longer paths between members, such as *graph partitions* and *bipartite cores*. All of these are discussed below.

Definition 21 (Clustering Coefficient). *For a node v with edges (u, v) and (v, w), the clustering coefficient of v measures the probability of existence of the third edge (u, w) (Figure 6.1(a)). The clustering coefficient of the entire graph is found by averaging over all nodes in the graph*[1].

[1]We note here that there is at least one other definition based on counting triangles in the graph; both definitions make sense, but we use this one.



(a) Clustering coefficient (b) Bipartite core

Figure 6.1: *Indicators of community structure:* (a) Node X has 6 neighbors. These neighbors could have been connected by $\binom{6}{2} = 15$ edges, but only 5 such edges exist. So, the local clustering coefficient of node X is $5/15 = 1/3$. (b) A 4×3 bipartite core, with each node in Set L being connected to each node in Set R.

Definition 22 (Graph Partitioning). *Graph partitioning techniques typically break the graph into two disjoint partitions (or communities) while optimizing some measure; these two communities may then be repartitioned separately.*

The popular `METIS` software tries to find the best separator, minimizing the number of edges cut in order to form two disconnected components of relatively similar sizes [Karypis and Kumar, 1998a]. Many other measures and techniques exist [Brandes et al., 2003, Alon, 1998]. Our Cross-associations method also finds such partitions, while optimizing the cost of encoding the adjacency matrix of the graph, as we have seen in Chapter 5.

Definition 23 (Bipartite Core). *A bipartite core in a graph consists of two (not necessarily disjoint) sets of nodes L and R such that every node in L links to every node in R; links from R to L are optional (Figure 6.1(b)).*

Significance of graph communities: Most real-world graphs exhibit strong community effects. Moody [Moody, 2001] found groupings based on race and age in a network of friendships in one American school, Schwartz and Wood [Schwartz and Wood, 1993] group people with shared interests from email logs, Borgs et al. [Borgs et al., 2004] find communities from "cross-posts" on Usenet, and Flake et al. [Flake et al., 2000] discover communities of webpages in the WWW. This is also reflected in the clustering coefficients of real-world graphs: they are almost always much larger than in random graphs of the same size [Watts and Strogatz, 1998].

Other Patterns:

Many other graph patterns have also been studied in the literature. We mention two of these that we will use later: (a) Edge-betweenness or Stress, (b) "Singular vector value" versus rank plots, and Resilience under attack.

Definition 24 (Edge betweenness or Stress). *Consider all shortest paths between all pairs of nodes in a graph. The edge-betweenness or stress of an edge is the number of these shortest paths that the edge belongs to, and is thus a measure of the "load" on that edge.*

Definition 25 (Stress Plot). *This is a plot of the number of nodes s_k with stress k, versus k.*

Definition 26 ("Singular vector value" versus rank plots). *The "singular vector value" of a node is the absolute value of the corresponding component of the first singular vector of the graph. It can be considered to be a measure of the "importance" of the node, and as we will see later, is closely related to the widely-used concept of "Bonacich centrality" in social network analysis [Bonacich, 1987].*

Definition 27 (Resilience). *A resilient graph is one whose diameter does not increase when its nodes or edges are removed according to some "attack" process [Palmer et al., 2002, Albert et al., 2000]. Most real-world graphs are very resilient against random failures, but susceptible to targeted attacks (such as removal of nodes of the highest degree) [Tangmunarunkit et al., 2001].*

6.1.2 Graph Generators

Graph generators allow us to create synthetic graphs, which can then be used for, say, simulation studies. However, to be realistic, the generated graph must match all (or at least several) of the patterns mentioned above. By telling us which processes can (or cannot) lead to the development of these patterns, graph generators can provide insight into the creation of real-world graphs.

Graph models and generators can be broadly classified into four categories:

1. *Random graph models:* The graphs are generated by a random process. The basic random graph model has attracted a lot of research interest due to its phase transition properties.

2. *Preferential attachment models:* In these models, the "rich" get "richer" as the network grows, leading to power law effects. Some of today's most popular models belong to this class.

3. *Optimization-based models:* Here, power laws are shown to evolve when risks are minimized using limited resources. Together with the preferential attachment models, they try to provide mechanisms that automatically lead to power laws.

4. *Geographical models:* These models consider the effects of geography (i.e., the *positions* of the nodes) on the growth and topology of the network. This is especially important for modeling router or power-grid networks, which involve laying wires between points on the globe.

We will briefly touch upon some of these generators below. Tables 6.1 and 6.2 provide a taxonomy of generators.

Random graph models:

In the earliest random graph model [Erdős and Rényi, 1960], we start with N nodes, and for every pair of nodes, an edge is added between them with probability p. This simple model leads to a surprising list of properties, including phase transitions in the size of the largest component, and

Generator	Graph type						Degree distributions			
							Power law			*Exponen-*
	Undir.	*Dir.*	*Bip.*	*Self loops*	*Mult. edges*	*Geog. info*	*Plain*	*Exp. cutoff*	*Devia-tion*	*tial*
Erdős–Rényi [Erdős and Rényi, 1960]	√			√	√					√
PLRG [Aiello et al., 2000], PLOD [Palmer and Steffan, 2000]	√			√	√		√			
Exponential cutoff [Newman et al., 2001]	√			√	√		√	√		
BA [Barabási and Albert, 1999]	√						√ ($\gamma = 3$)			
AB [Albert and Barabási, 2000]	√			√	√		√			√
Edge Copying [Kleinberg et al., 1999], [Kumar et al., 1999]		√		√	√		√			
GLP [Bu and Towsley, 2002]	√			√	√		√			
Accelerated growth [Barabási et al., 2002]	√			√	√		Power-law mixture of $\gamma = 2$ and $\gamma = 3$			
Fitness model [Bianconi and Barabási, 2001]	√						√ (modified)			
Aiello et al. [Aiello et al., 2001]		√					√			
Pandurangan et al. [Pandurangan et al., 2002]		√		√	√		√			
Inet [Winick and Jamin, 2002]	√						√	√		
Pennock et al. [Pennock et al., 2002]	√			√	√		√		√	
Small-world [Watts and Strogatz, 1998]	√					√				√
Waxman [Waxman, 1988]	√					√		√		
BRITE [Medina et al., 2000]	√					√	√			
Yook et al. [Yook et al., 2002]	√					√	√		√	
Fabrikant et al. [Fabrikant et al., 2002]	√					√	√			
R-MAT [Chakrabarti et al., 2004]	√	√	√	√	√		√		√ (DGX)	

Table 6.1: *Taxonomy of graph generators*: This table shows the graph types and degree distributions that different graph generators can create. The graph type can be undirected, directed, bipartite, allowing self-loops or multi-graph (multiple edges possible between nodes). The degree distributions can be power-law (with possible exponential cutoffs, or other deviations such as log-normal/DGX) or exponential decay. Empty cells indicate that the corresponding property does not occur in the corresponding model.

Generator	Diameter or Avg path len.	Community		Clustering coefficient	Remarks
		Bip. core vs size	$C(k)$ *vs* k		
Erdős–Rényi [Erdős and Rényi, 1960]	$O(\log N)$		Indep.	Low, $CC \propto N^{-1}$	
PLRG [Aiello et al., 2000], PLOD [Palmer and Steffan, 2000]	$O(\log N)$	Indep.		$CC \to 0$ for large N	
Exponential cutoff [Newman et al., 2001]	$O(\log N)$			$CC \to 0$ for large N	
BA [Barabási and Albert, 1999]	$O(\log N)$ or $O(\frac{\log N}{\log \log N})$			$CC \propto N^{-0.75}$	
AB [Albert and Barabási, 2000]					
Edge copying [Kleinberg et al., 1999], [Kumar et al., 1999]		Power-law			
GLP [Bu and Towsley, 2002]				Higher than AB, BA, PLRG	Internet only
Accelerated growth [Barabási et al., 2002]				Non-monotonic with N	
Fitness model [Bianconi and Barabási, 2001]					
Aiello et al. [Aiello et al., 2001]					
Pandurangan et al. [Pandurangan et al., 2002]					
Inet [Winick and Jamin, 2002]					Specific to the AS graph
Pennock et al. [Pennock et al., 2002]					
Small-world [Watts and Strogatz, 1998]	$O(N)$ for small N, $O(\ln N)$ for large N, depends on p			$CC(p) \propto (1-p)^3$, Indep of N	N=num nodes p=rewiring prob
Waxman [Waxman, 1988]					
BRITE [Medina et al., 2000]	Low (like in BA)			like in BA	BA + Waxman with additions
Yook et al. [Yook et al., 2002]					
Fabrikant et al. [Fabrikant et al., 2002]					Tree, density 1
R-MAT [Chakrabarti et al., 2004]	Low (empirically)				

Table 6.2: *Taxonomy of graph generators (Contd.)*: The comparisons are made for graph diameter, existence of community structure (number of bipartite cores versus core size, or Clustering coefficient $CC(k)$ of all nodes with degree k versus k), and clustering coefficient. N is the number of nodes in the graph. The empty cells represent information unknown to the authors, and require further research.

diameter logarithmic in graph size. Its ease of analysis has proven to be very useful in the early development of the field. However, its degree distribution is Poisson, whereas most real-world graphs seem to exhibit power law distributions. Also, the graphs lack community effects: the clustering coefficient is usually far smaller than that in comparable real-world graphs.

The basic Erdős-Rényi model has been extended in several ways, typically to match the power law degree distribution pattern [Aiello et al., 2000, Palmer and Steffan, 2000, Newman et al., 2001]. These methods retain the simplicity and ease of analysis of the original model, while removing one of its weaknesses: the unrealistic degree distribution. However, these models do not describe any *process* by which power laws may arise automatically, which makes them less useful in understanding the internal processes behind graph formation in the real world (property **(P2)**). Also, most models make no attempt to match any other patterns (property **(P1)**), and further work is needed to incorporate community effects into the model.

Preferential attachment models:

First developed in the mid-1950s [Simon, 1955], the idea of preferential attachment has been re-discovered recently due to their ability to generate skewed distributions such as power laws [Price, 1976, Barabási and Albert, 1999]. Informally, they use the concept of the "rich getting richer" over time: new nodes join the graph each timestep, and preferentially connect to existing nodes with high degree. This basic idea has been very influential, and has formed the basis of a large body of further work [Albert and Barabási, 2000, Bu and Towsley, 2002, Barabási et al., 2002, Bianconi and Barabási, 2001, Aiello et al., 2001, Bollobás et al., 2003, Pandurangan et al., 2002, Winick and Jamin, 2002, Pennock et al., 2002, Albert and Barabási, 2002].

The preferential attachment models have several interesting properties:

- *Power law degree distributions:* These models lead to power laws as a *by-product* of the graph generation method, and not as a specific designed-in feature.
- *Low diameter:* The generated graphs have $O(\log N)$ diameter. Thus, the increase in diameter is slower than the growth in graph size.
- *Resilience:* The generated graphs are resilient against random node/edge removals, but quickly become disconnected when nodes are removed in descending order of degree [Albert et al., 2000, Palmer et al., 2002]. This matches the behavior of the Internet.
- *A procedural method:* Perhaps most importantly, these models describe a *process* that can lead to realistic graphs and power laws (matching property **(P2)**). The two main ideas are those of (a) growth, and (b) preferential attachment; variants to the basic model add other ideas such as node "fitness." The ability of these models to match many real-world graphs implies that graphs in the real-world might indeed have been generated by similar processes.

Preferential attachment models are probably the most popular models currently, due to their ability to match power law degree distributions by such a simple set of steps. However, they typically do not exhibit community effects, and, apart from the paper of Pennock et al. [Pennock

et al., 2002], little effort has gone into finding reasons for deviations from power laws in real-world graphs (property **(P1)**).

One set of related models has shown promise recently: these are the "edge copying" models [Kleinberg et al., 1999, Kumar et al., 1999], where a node (such as a website) acquires edges by *copying links* from other nodes (websites). This is similar to preferential attachment because pages with high-degree will be linked to by many other pages, and so have a greater chance of getting copied. However, such graphs can be expected to have a large number of bipartite cores (which leads to the community effect). This makes the edge-copying technique a promising research direction.

Geographical models:

Several models introduce the constraints of geography into network formation. For example, it is easier (cheaper) to link two routers which are physically close to each other; most of our social contacts are people we meet often, and who consequently probably live close to us (say, in the same town or city), and so on. In this area, there are two important models: the *Small-World* model, and the *Waxman* model.

The *Small-World* model [Watts and Strogatz, 1998] starts with a regular lattice and adds/rewires some edges randomly. The original lattice represents ties between close friends, while the random edges link "acquaintances," and serve to connect different social circles. Thus, the resulting graph has low diameter but a high clustering coefficient — two patterns common to real-world graphs. However, the degree distribution is not a power law, and the basic model needs extensions to match this (property **(P1)**).

The *Waxman* model [Waxman, 1988] probabilistically links two nodes in the graph, based on their geographical distance (in fact, the probability decreases exponentially with distance). The model is simple yet attractive, and has been incorporated into the popular BRITE [Medina et al., 2000] generator used for graph generation in the networking community. However, it does not yield a power law degree distribution, and further work is needed to analyze the other graph patterns for this generator (property **(P1)**).

Optimization-based models:

Optimization-based models provide another process leading to power laws. Carlson and Doyle [Carlson and Doyle, 1999, Doyle and Carlson, 2000] propose in their *Highly Optimized Tolerance (HOT)* model that power laws may arise in systems due to *tradeoffs* between yield (or profit), resources (to prevent a risk from causing damage) and tolerance to risks. Apart from power laws, HOT also matches the resilience pattern of many real-world graphs (see Definition 27): it is robust against "designed-for" uncertainties, but very sensitive to design flaws and unanticipated perturbations.

Several variations of the basic model have also been proposed: COLD [Newman et al., 2002] truncates the power law tails, while the *Heuristically Optimized Tradeoffs* model [Fabrikant et al., 2002] needs only locally-optimal decisions instead of global optimality.

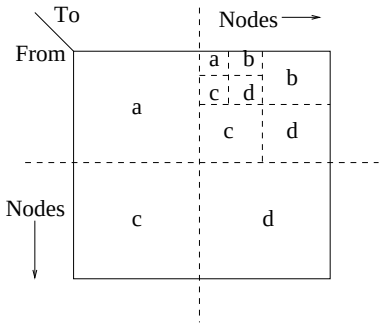


Figure 6.2: *The R-MAT model:* The adjacency matrix is broken into four equal-sized partitions, and one of those four is chosen according to a (possibly non-uniform) probability distribution. This partition is then split recursively till we reach a single cell, where an edge is placed. Multiple such edge placements are used to generate the full synthetic graph.

Optimization-based models provide a very intuitive process which leads to both power laws and resilience (matching property **(P2)**). However, further research needs to be conducted on other patterns for such graphs (property **(P1)**). One step in this direction is the work of Berger et al. [Berger et al., 2005], who generalize the *Heuristically Optimized Tradeoffs* model, and show that it is equivalent to a form of preferential attachment; thus, competition between opposing forces can give rise to preferential attachment, and we already know that preferential attachment can, in turn, lead to power laws and exponential cutoffs.

Summary of graph generators

Thus, we see that these are four general categories of graph generators, along with several "cross-category" generators too. Most of the generators do well with properties **(P3)**-**(P6)**, but need further research to determine their realism (property **(P1)**, that is, which patterns they match, and which they don't). Next, we will discuss our R-MAT graph generator, which attempts to match all of these properties, including the degree distribution (with both power laws and deviations), community structure, singular vector value versus rank patterns and so on.

6.2 The R-MAT methodology

We have seen that most of the current graph generators focus on only one graph pattern – typically the degree distribution – and give low importance to all the others. There is also the question of how to fit model parameters to match a given graph. What we would like is a tradeoff between parsimony (property **(P3)**), realism (property **(P1)**), and efficiency (properties **(P4)** and **(P5)**). In this section, we present the R-MAT generator, which attempts to address all of these concerns.

Description and properties:

R-MAT is based on the well-known "$80-20$ rule" in one dimension (80% of the data falls on 20% of the data range), which is known to result in self-similarity and power laws; R-MAT

extends this rule to a two-dimensional graph adjacency matrix. The R-MAT generator creates directed graphs with 2^n nodes and E edges, where both values are provided by the user. We start with an empty adjacency matrix, and divide it into four equal-sized partitions. One of the four partitions is chosen with probabilities a, b, c, d respectively ($a + b + c + d = 1$), as in Figure 6.2. The chosen partition is again subdivided into four smaller partitions, and the procedure is repeated until we reach a simple cell (=1×1 partition). The nodes (that is, row and column) corresponding to this cell are linked by an edge in the graph. This process is repeated E times to generate the full graph. There is a subtle point here: we may have *duplicate* edges (i.e., edges which fall into the same cell in the adjacency matrix), but we only keep one of them when generating an unweighted graph. To smooth out fluctuations in the degree distributions, some noise is added to the (a, b, c, d) values at each stage of the recursion, followed by renormalization (so that $a + b + c + d = 1$). Typically, $a \geq b$, $a \geq c$, $a \geq d$.

Parsimony: The algorithm needs only three parameters: the partition probabilities a, b, and c; $d = 1 - a - b - c$. Thus, the models is parsimonious.

Degree distribution: The following theorem gives the expected degree distribution of an R-MAT generated graph.

Theorem 4 (Count-vs-degree). *For a pure R-MAT generated graph (ie., without any smoothing factors), the expected number of nodes c_k with outdegree k is given by*

$$c_k = \binom{E}{k} \sum_{i=0}^{n} \binom{n}{i} \left[p^{n-i}(1-p)^i\right]^k * \left[1 - p^{n-i}(1-p)^i\right]^{E-k} \tag{6.2}$$

where 2^n is the number of nodes in the R-MAT graph (typically $n = \lceil \log_2 N \rceil$ and $p = a + b$.

Proof. Proved in Section 6.4. □

This is well modeled by a *discrete lognormal* [Bi et al., 2001], which looks like a truncated parabola on the log-log scale. By setting the parameters properly, this can successfully match both power-law and "unimodal" distributions [Pennock et al., 2002].

Communities: Intuitively, R-MAT is generating "communities" in the graph:

- The partitions a and d represent separate groups of nodes which correspond to communities (say, "Linux" and "Windows" users).
- The partitions b and c are the *cross-links* between these two groups; edges there would denote friends with separate preferences.
- The recursive nature of the partitions means that we automatically get sub-communities within existing communities (say, "RedHat" and "Mandrake" enthusiasts within the "Linux" group).

Diameter, singular values and other properties: We show experimentally that graphs generated by R-MAT have small diameter and match several other criteria as well.

Extensions to undirected, bipartite and weighted graphs: The basic model generates directed graphs; all the other types of graphs can be easily generated by minor modifications of the model. For undirected graphs, a directed graph is generated and then made symmetric. For bipartite graphs, the same approach is used; the only difference is that the adjacency matrix is now rectangular instead of square. For weighted graphs, the number of *duplicate* edges in each cell of the adjacency matrix is taken to be the weight of that edge. More details may be found in [Chakrabarti et al., 2004].

Parameter fitting algorithm: We are given some input graph, and need to fit the R-MAT model parameters so that the generated graph matches the input graph in terms of graph patterns. Using Theorem 4 we can fit the indegree and outdegree distributions; this gives us two equations (specifically, we get the values of $p = a + b$ and $q = a + c$). We need one more equation to fit the three model parameters.

We tried several experiments where we fit the *scree plot* (see Definition 18). However, we obtained comparable (and much faster) results by conjecturing that the $a : b$ and $a : c$ ratios are approximately $75 : 25$ (as seen in many real world scenarios), and using these to fit the parameters. Hence, this is our current parameter-fitting method for R-MAT.

Relationship with Cross-Associations: In Chapter 5, we described our Cross-Associations algorithm to extract "natural" communities in graphs, and R-MAT specifically generates graphs with a recursive community structure. The similarity between these two, however, is not very deep. While R-MAT creates a community "hierarchy," Cross-Associations finds only a flat set of communities, which need not strictly correspond to the R-MAT communities at any level of the hierarchy. The real linkage is in the *sizes* of the communities found: when the R-MAT parameters are very skewed (say, $a \gg d$), the communities found by Cross-Associations have very skewed sizes, with several very large sparse communities and a few extremely small densely-connected communities. Conversely, when the R-MAT parameters are more equal, the community sizes are more similar to each other.

Relationship with tree-based generators: The process of placing edges during R-MAT graph generation can be thought of as a tree traversal from root to leaf: each internal node of this tree has four children (one for each "quadrant" in R-MAT), the leaves of the tree are the individual cells of the adjacency matrix, and each "choosing of a quadrant" in R-MAT corresponds to a descent of one level in this tree. This is similar in spirit to other tree-based graph generators [Kleinberg, 2001, Watts et al., 2002] which use hierarchies in forming links between nodes; however, the differences are significant. The R-MAT tree has all possible *edges* as its leaves, while the hierarchy-based generators have *nodes* as leaves. The latter use a tree-distance metric to compute the probability of an edge, but the usefulness of tree-distance in the R-MAT tree is not clear. (What does the tree-distance between two edges mean?) The R-MAT tree automatically generates the realistic degree distribution described above, while for the other tree-based methods, the degree distribution also depends on how many edges we attach to each node; hence, a large number of free parameters need to be set. Thus, while R-MAT is conceptually related to these tree-based methods, it is also significantly different.

6.3 Experiments

We show experiments on the following graphs:

- The *EPINIONS* directed graph: A graph of who-trusts-whom from epinions.com [Richardson and Domingos, 2002]: $N = 75,879; E = 508,960$.
- The *OREGON* directed graph: A graph of connections between Internet Autonomous Systems, obtained from `http://topology.eecs.umich.edu/data.html`: $N = 11,461; E = 32,730$.
- The *CITATIONS* directed graph: A graph of paper connections, obtained from the KDD Cup (2003) website: `http://www.cs.cornell.edu/projects/kddcup/datasets.html`: $N = 25,059; E = 352,807$.
- The *CLICKSTREAM* bipartite graph: A graph of the browsing behavior of Internet users [Montgomery and Faloutsos, 2001]. An edge (u, p) denotes that user u accessed page p. It has $23,396$ users, $199,308$ pages and $952,580$ edges.
- The *EPINIONS-U* undirected graph: This is an undirected version of the *EPINIONS* graph: $N = 75,879; E = 811,602$.

The questions we want to answer are:

- **(Q1)** How well does R-MAT match directed graphs, such *EPINIONS*, *OREGON* and *CITATIONS*?
- **(Q2)** How well does R-MAT match bipartite graphs, such as *CLICKSTREAM*?
- **(Q3)** How well does R-MAT match undirected graphs, such as *EPINIONS-U*?

For each of the datasets, we fit the R-MAT parameters, and compare the true graph with one generated by R-MAT. We also compare R-MAT with three existing generators chosen for their popularity or recency:

- The *AB* model [Albert and Barabási, 2000]: This is a preferential attachment model with an additional process to rewire edges and add links between existing nodes (instead of only adding links to new nodes).
- The *Generalized Linear Preference (GLP)* model [Bu and Towsley, 2002]: This modifies the original preferential attachment equation with an extra parameter, and is highly regarded in the networking community.
- The *PG* model [Pennock et al., 2002]: This model has a parameter to traverse the continuum from pure preferential attachment to pure random attachment.

There are two important points to note:

Dataset	Graph type	Dimensions	Edges	R-MAT parameters			
				a	b	c	d
EPINIONS	Directed	$75,879 \times 75,879$	$508,960$	0.56	0.19	0.18	0.07
OREGON	Directed	$11,461 \times 11,461$	$32,730$	0.597	0.165	0.233	0.005
CITATIONS	Directed	$25,059 \times 25,059$	$352,807$	0.538	0.111	0.248	0.103
CLICKSTREAM	Bipartite	$23,396 \times 199,308$	$952,580$	0.50	0.15	0.19	0.16
EPINIONS-U	Undirected	$75,879 \times 75,879$	$811,602$	0.55	0.18	0.18	0.09

Table 6.3: *R-MAT parameters for the datasets*

- All of the above models are used to generate undirected graphs, and thus, we can compare them to R-MAT only on *EPINIONS-U*.
- We were unaware of any method to fit the parameters of these models, so we fit them using a brute-force method. We use *AB+*, *PG+* and *GLP+* for the original algorithms augmented by our parameter fitting.

The graph patterns we look at are:

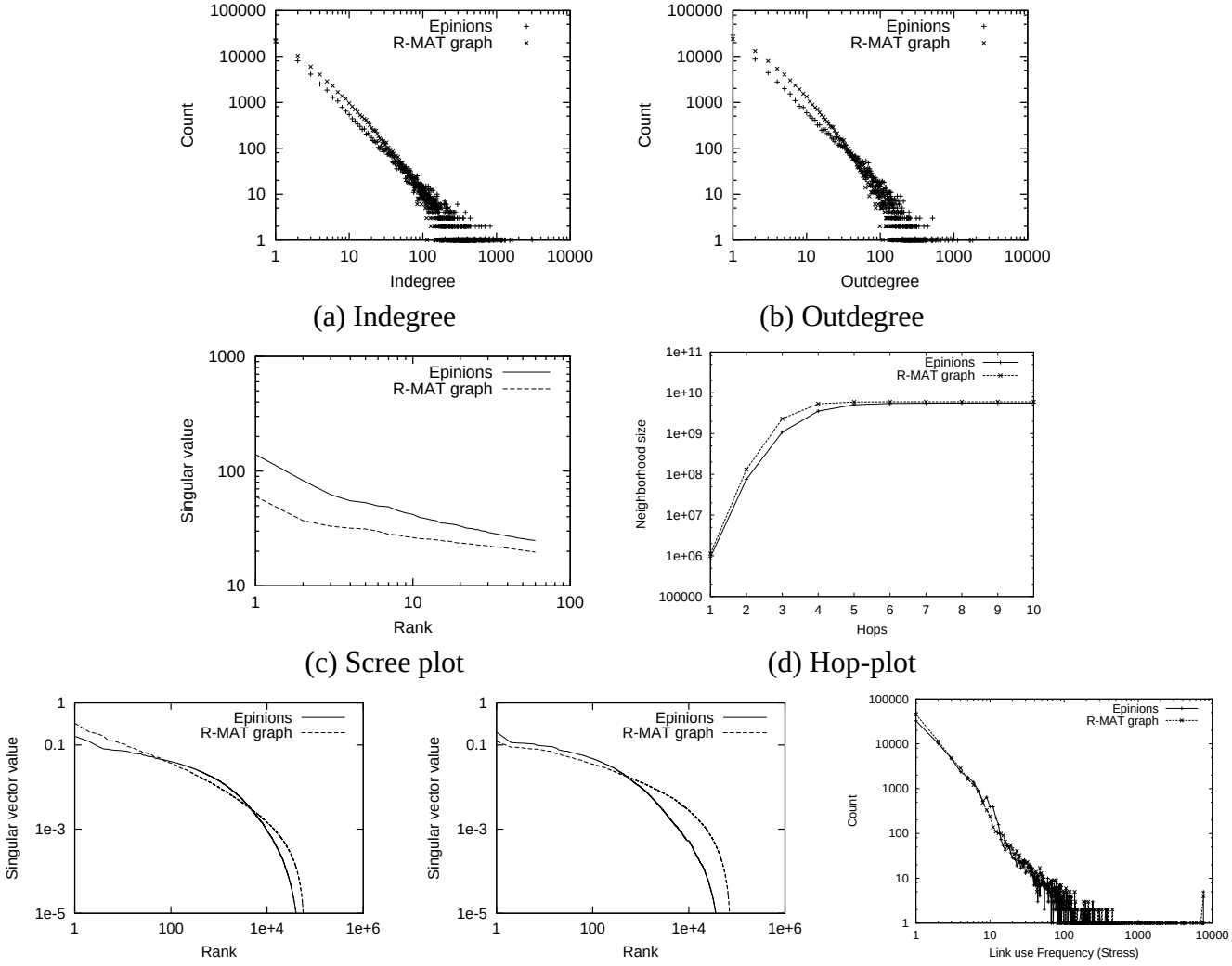
1. Both indegree and outdegree distributions (Definition 17).
2. "Hop-plot" and "effective diameter" (Definitions 19 and 20).
3. Singular value vs. rank plot (also known as the *scree plot*; see Definition 18).
4. "Singular vector value" versus rank plots (Definition 26).
5. "Stress" distribution (Definition 25).

6.3.1 (Q1) R-MAT on Directed Graphs

Figures 6.3, 6.4 and 6.5 show results on the *EPINIONS*, *OREGON* and *CITATIONS* directed graphs. The R-MAT fit is very good; the other models considered are not applicable. The corresponding R-MAT parameters are shown in Table 6.3.

6.3.2 (Q2) R-MAT on Bipartite Graphs

Figure 6.6 shows results on the *CLICKSTREAM* bipartite graph. As before, the R-MAT fit is very good. In particular, note that the indegree distribution is a power law while the outdegree distribution deviates significantly from a power law; R-MAT matches *both* of these very well. Again, the other models are not applicable.



(a) Indegree (b) Outdegree

(c) Scree plot (d) Hop-plot

(e) First left singular vector (f) First right singular vector (g) Stress

Figure 6.3: *Results on the* EPINIONS *directed graph:* The *AB+*, *PG+* and *GLP+* methods **do not apply**. The crosses and dashed lines represent the R-MAT generated graphs, while the pluses and strong lines represent the real graph.

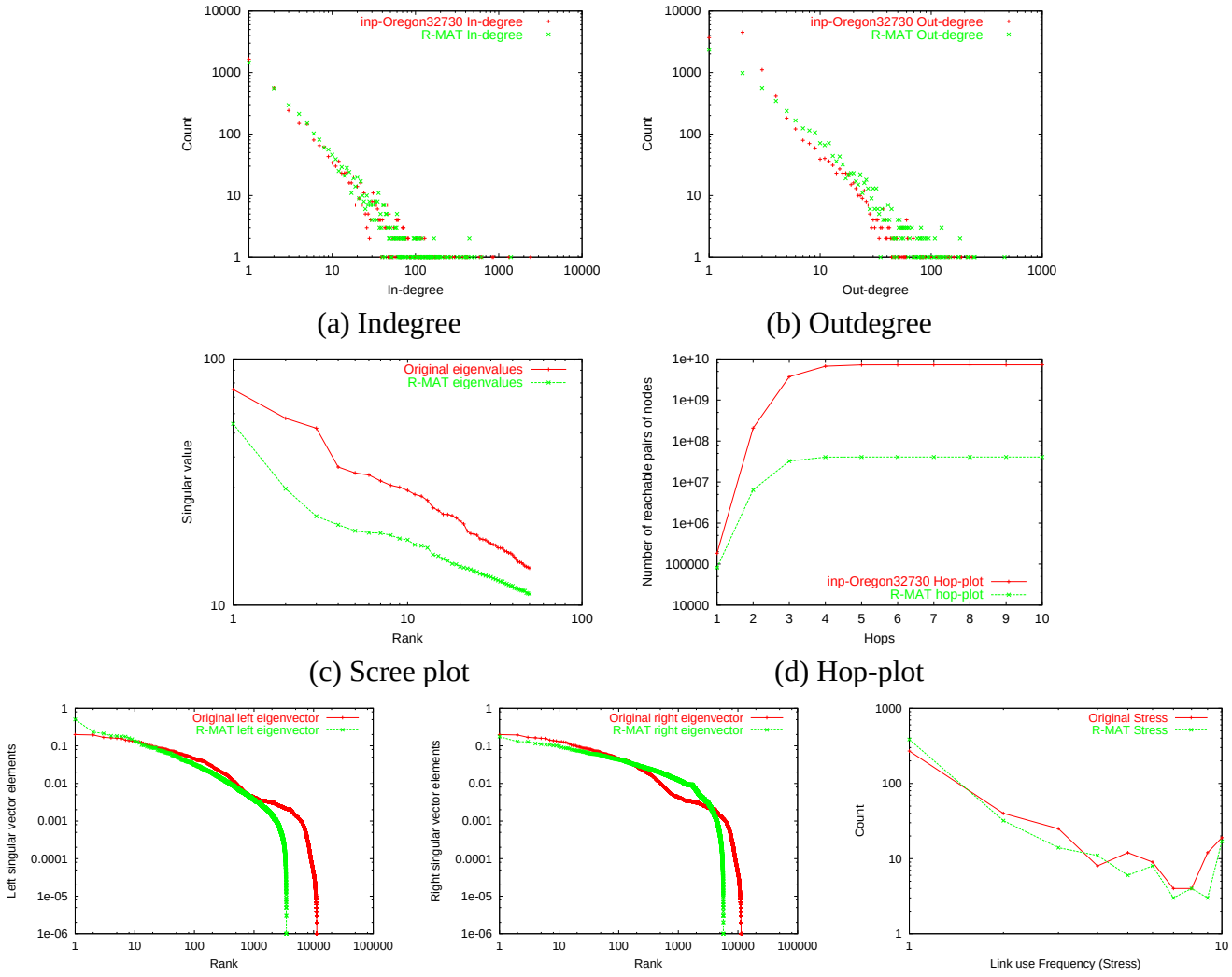


Figure 6.4: *Results on the* OREGON *directed graph:* The *AB+*, *PG+* and *GLP+* methods **do not apply**. The crosses and dashed lines represent the R-MAT generated graphs, while the pluses and strong lines represent the real graph.

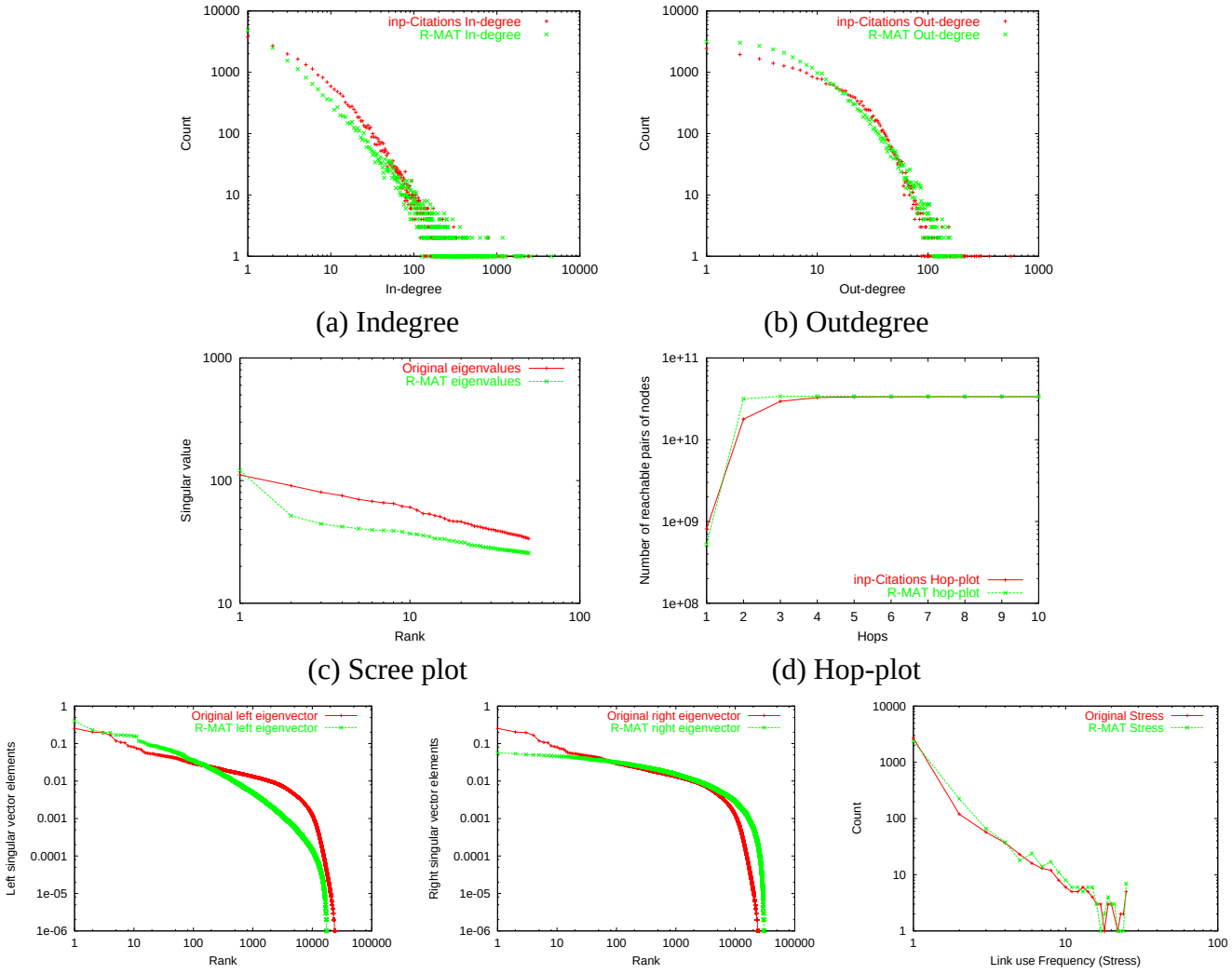
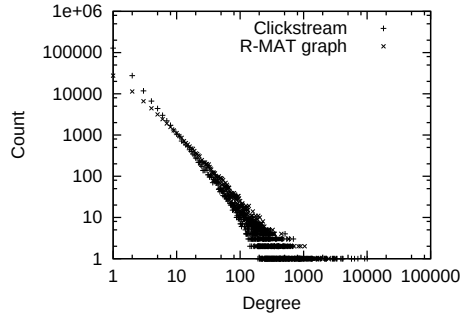
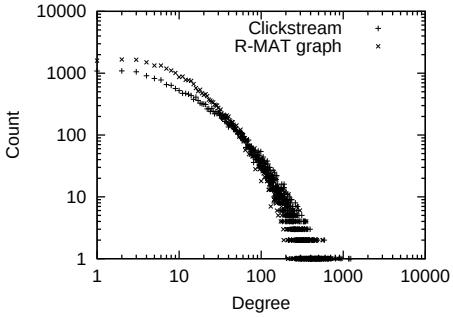


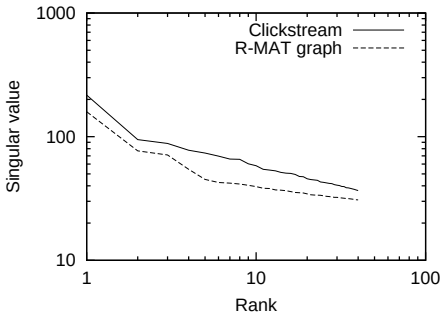
Figure 6.5: *Results on the* CITATIONS *directed graph:* The *AB+*, *PG+* and *GLP+* methods **do not apply**. The crosses and dashed lines represent the R-MAT generated graphs, while the pluses and strong lines represent the real graph.



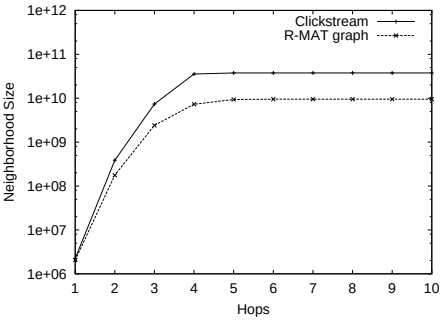
(a) Indegree



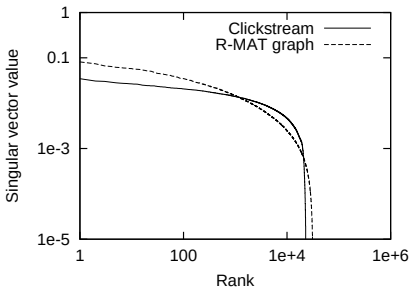
(b) Outdegree



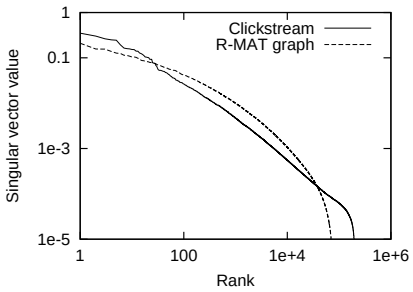
(c) Scree plot



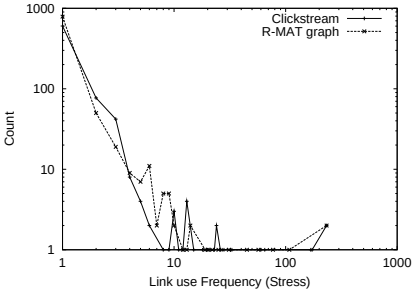
(d) Hop-plot



(d) First left singular vector

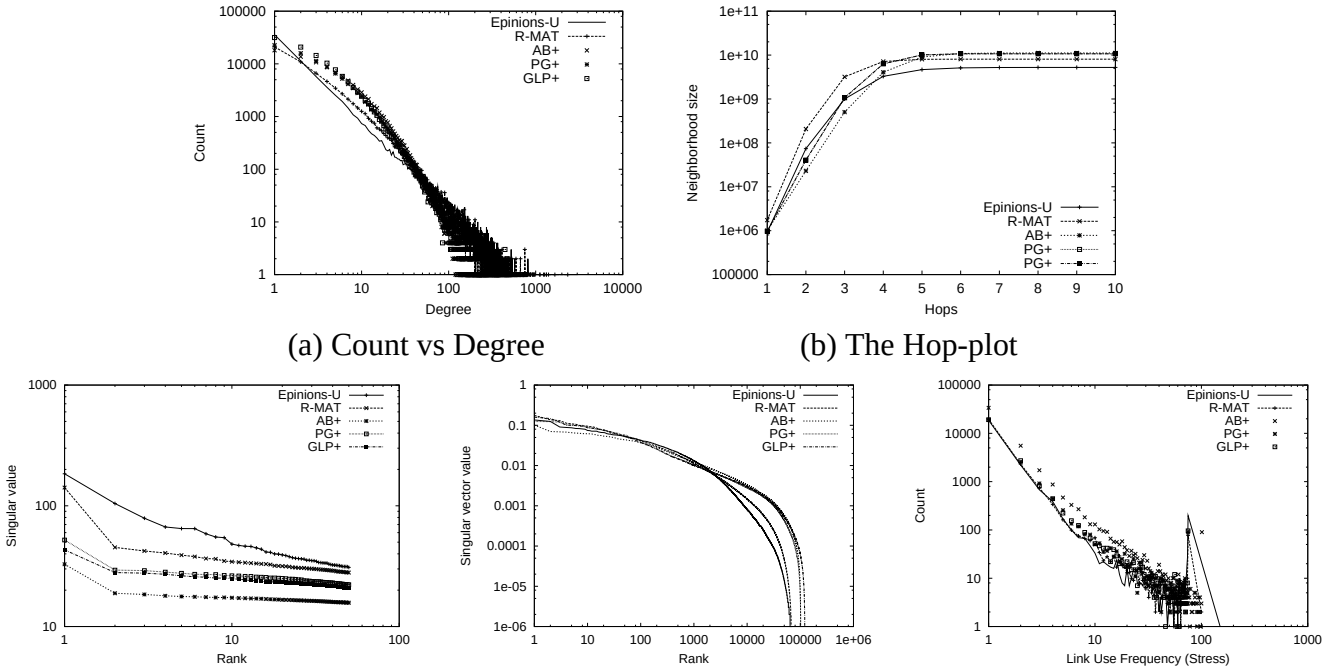


(e) First right singular vector



(g) Stress

Figure 6.6: *Results on the* CLICKSTREAM *bipartite graph:* The *AB+*, *PG+* and *GLP+* methods **do not apply**. The crosses and dashed lines represent the R-MAT generated graphs, while the pluses and strong lines represent the real graph.



(a) Count vs Degree (b) The Hop-plot

(c) Singular value vs Rank (d) Singular vector vs Rank (e) Stress

Figure 6.7: *Results on the* EPINIONS-U *undirected graph:* We show (a) degree, (b) hop-plot, (c) singular value, (d) "singular vector value," and (e) stress distributions for the *EPINIONS-U* dataset. R-MAT gives the best matches to the *EPINIONS-U* graph, among all the generators. In fact, for the stress distribution, the R-MAT and *EPINIONS-U* plots are almost indistinguishable.

6.3.3 (Q3) R-MAT on Undirected Graphs

Figure 6.7 shows the comparison plots on the *EPINIONS-U* undirected graph. R-MAT gives the closest fits. Also, note that all the y-scales are logarithmic, so small differences in the plots actually represent significant deviations.

6.4 Details of Proofs

Theorem 1 (Count-vs-degree). *For a pure R-MAT generated graph (ie., without any smoothing factors), the expected number of nodes c_k with outdegree k is given by*

$$c_k = \binom{E}{k} \sum_{i=0}^{n} \binom{n}{i} \left[p^{n-i}(1-p)^i\right]^k * \left[1 - p^{n-i}(1-p)^i\right]^{E-k} \tag{6.3}$$

where 2^n is the number of nodes in the R-MAT graph and $p = a + b$.

Proof. In the following discussion, we neglect the elimination of duplicate edges. This is a reasonable assumption: in most of our experiments, we found that the number of duplicate edges is far less than the total number of edges. Each edge that is "dropped" on to the adjacency matrix

takes a specific path: at each stage of the recursion, the edge chooses either Up (corresponding to partitions a or b) or Down (corresponding to partitions c or d). There are n such stages, where 2^n is the number of nodes. Row X can be reached only if the edge follows a *unique* ordered sequence of Up and Down choices. Since the probability of choosing Up is $p = a + b$ and the probability of choosing Down is $1 - p = c + d$, the probability of the edge falling to row X is

$$\begin{aligned} P(X) &= p^{\text{num(Up)}}.(1-p)^{\text{num(Down)}} \\ &= p^{\text{num(Up)}}.(1-p)^{n-\text{num(Up)}} \end{aligned} \tag{6.4}$$

This equation means that any other row which requires the same number of Up choices will have the same probability as row X. The number of such rows can be easily seen to be $\binom{n}{\text{num(Up)}}$. Thus, we can think of different *classes* of rows:

Class	Probability of getting an edge	Num(rows)
0	p^n	$\binom{n}{0}$
1	$p^{n-1}(1-p)^1$	$\binom{n}{1}$
$\vdots$	$\vdots$	$\vdots$
i-1	$p^{n-i}(1-p)^i$	$\binom{n}{i}$
$\vdots$	$\vdots$	$\vdots$
n	$(1-p)^n$	$\binom{n}{n}$

Now, we can calculate the count-degree plot. Let

$$\begin{aligned} NR_k &= \text{number of rows with outdegree } k \\ &= NR_{0,k} + NR_{1,k} + \ldots + NR_{n,k} \end{aligned} \tag{6.5}$$

where $NR_{i,k} =$ number of rows *of Class* i with outdegree k. Thus, the expected number of rows with outdegree k is:

$$E[NR_k] = \sum_{i=0}^{n} E[NR_{i,k}] \tag{6.6}$$

Now, for a row in Class i, each of the E edges can either drop into it (with probability $p^{n-i}(1-p)^i$) or not (with probability $1 - p^{n-i}(1-p)^i$). Thus, the number of edges falling into this row is a binomially distributed random variable: $Bin(E, p^{n-i}(1-p)^i)$. Thus, the probability that it has exactly k edges is given by

$$P_{i,k} = \binom{E}{k} \left[p^{n-i}(1-p)^i\right]^k \left[1 - p^{n-i}(1-p)^i\right]^{E-k}$$

Thus, the expected number of such rows from Class i is

$$\begin{aligned} &E[NR_{i,k}] = \text{number of rows in Class } i * P_{i,k} \\ &= \binom{n}{i} * P_{i,k} \\ &= \binom{n}{i}\binom{E}{k} \left[p^{n-i}(1-p)^i\right]^k \\ &\qquad * \left[1 - p^{n-i}(1-p)^i\right]^{E-k} \end{aligned} \tag{6.7}$$

Using this in Equation 6.5 gives us:

$$E[NR_k] = \binom{E}{k} \sum_{i=0}^{n} \binom{n}{i} \left[p^{n-i}(1-p)^i\right]^k * \left[1 - p^{n-i}(1-p)^i\right]^{E-k} \quad (6.8)$$

Equation 6.8 gives us the count of nodes with outdegree k; thus we can plot the count-vs-outdegree plot using this equation. □

6.5 Summary

We presented the R-MAT graph generator, a simple parsimonious model for efficiently generating realistic graphs. Instead of focusing on just one pattern like the degree distributions (as most current generators do), R-MAT attempts to match several patterns, including diameter and hop-plot, the scree plot of singular values versus rank, "stress" plots and others. We experimentally demonstrate our parameter-fitting algorithm, and the ability of R-MAT to successfully mimic the patterns found in several real-world graphs. This opens up the possibility of using R-MAT for many applications: generating small samples "similar" to a large given graph, or for simulation studies when a "true" real-world graph may not exist yet, or may be too costly to obtain, or for use in graph compression algorithms.

Chapter 7

Conclusions

Graphs are ubiquitous; they show up in fields as diverse as ecological studies, sociology, computer networking and many others. In fact, any information relating different entities (an $M : N$ relationship in database terminology) can be thought of as a graph. Mining the hidden patterns in graphs helps define what is "normal" for real-world graphs, and deviations from such patterns can imply abnormal graphs/subgraphs.

There is a dichotomy in graph mining applications: we can answer specific queries on a particular graph, or we can ask questions pertaining to real-world graphs in general. In my thesis, I explored issues from both sides of this dichotomy.

- *Problems for specific graphs:*
  - How does a virus propagate over the given graph? When does a viral outbreak die out?
  - Under what conditions does a piece of information survive in a given sensor network with failing nodes and links?
  - How can we automatically cluster nodes in a given graph? How can we quickly and automatically estimate the *number* of clusters in the data?
- *Real-world graphs in general:*
  - What patterns and "laws" hold for most real-world graphs? How can we *generate* synthetic yet "realistic" graphs?

Viral propagation: A quantitative and analytic understanding of the propagation behavior of viruses is critically important for many problems, including:

- the design of new anti-virus measures that combat the virus *globally*, that is, over the entire network,
- the determination of "susceptibility" of a computer (or social) network to viral infections,
- the design of new "virus-resistant" networks, and

- the "optimal" immunization strategies to best prevent the spread of a virus, and avoid an epidemic.

In addition, this work is not limited to viruses: the same principles also apply to the spread of rumors and fashions, to viral marketing, to information dissemination campaigns, and the like.

We formulated the virus propagation problem as a non-linear dynamical system (called *NLDS*), which requires is linear in the size of the network, and is tractable using analytic techniques. Our main contributions, apart from the design of *NLDS*, are:

- *Accuracy of* NLDS*:* We show that the number of infected nodes in the graph for *NLDS* is very close to that for the full Markov chain, for a variety of synthetic and real-world graphs. Thus, our approximation is very accurate.
- *The Epidemic Threshold:* We derived the epidemic threshold below which a viral outbreak is expected to die out, but above which it might survive for long. Surprisingly, this condition depends only on one property of the graph: its largest eigenvalue.

Information survival threshold: The problem of information survival in sensor networks is similar, but the occasional failure of nodes in the network adds more complexity. Our contributions are:

- *Problem formulation methodology:* We again demonstrated the formulation of the problem as a non-linear dynamical system, thus proving the generality of this approach.
- *The Information Survival Threshold:* Solving the non-linear dynamical system allows us to find the information survival threshold for any given sensor or P2P network. This is found to depend on the largest value of a properly constructed matrix, which links together the link qualities and failure rates in the sensor network.

As sensor networks become more and more commonplace, this result will be important in determining the "informational load" we can place on the network.

Automatic clustering of nodes:

Clustering the nodes in a graph is a powerful way to mine the data in a graph. It tells us the major "classes" of nodes, and their behaviors (which other "classes" do they link to strongly). The clusters can be used in visualization tools to quickly provide a summary of a large graph dataset. Clustering has applications in personalization services, consumer segmentation, fraud detection, collaborative filtering; the list goes on.

We developed an efficient and accurate method of finding clusters in large graphs, with many useful properties:

- *Completely automatic:* Our clustering method is one of the few that needs *no "magic" numbers.* It automatically figures out both the *number* of clusters in the data, and their *memberships*.
- *Scalable:* The method is linear in the number of edges in the graph, and is thus scalable to large graphs.

- *Incremental:* New data can easily be incorporated into the current clustering solution, thus avoiding the costs of full recomputation.
- *Applicability to both self- and bipartite-graphs:* The algorithm can be used for, say, both customer-product data as well as social-network data.
- *Outlier detection:* In addition to clustering, we also proposed efficient methods to *detect outlier edges* in the data, and *rank* them in decreasing order of "outlierness."
- *Inter-cluster "distances:"* An intuitive metric was proposed as a "distance" between clusters, and an efficient algorithm to compute it was developed.

The R-MAT graph generator:

A good graph generator is an essential part of a graph miner's toolbox. The generator must be able to match the common graph patterns and "laws," besides being efficient, parsimonious, and having a simple parameter-fitting algorithm. The uses are many:

- *Simulation studies:* Algorithms can be tested on synthetic yet realistic graphs, where obtaining real-world graphs might be costly or even impossible.
- *Outlier detection:* Deviations from the common patterns can signify outliers, which need to be investigated further.
- *Realism of samples:* Samples of graphs can be tested for the graph patterns to verify their realism.
- *Extrapolation of data:* The generator can be used in "what-if" scenarios, to visualize what current graphs would look like when they grow by $x\%$.

Our R-MAT graph generator is exactly is a step in this direction: it matches many of the graph patterns, while requiring only three parameters; the generation process is efficient and scalable; and, the same basic method can be used to generate weighted or unweighted, directed or undirected, self- and bipartite graphs. Indeed, we experimentally demonstrated the ability of R-MAT to fit several large real-world datasets of different types.

To conclude, we developed tools for mining graph datasets under a variety of circumstances, from the propagation characteristics of real-world graphs, to the clusters hidden within them, and finally on to the patterns and laws that govern them, and a generator to mimic these. Each of these is an important tool in its own right; combined together, their usefulness is increased even more. "How susceptible will the Internet be to viral infections if its grows by $x\%$ nodes and $y\%$ edges?" Our work can shed some light.

Chapter 8

Future Work

Graph mining is still a very young discipline. Even though parts of it have been studied in several communities for a while, many important questions are unanswered as yet. There are many avenues for future work, some of which will be discussed below.

Mining network traffic matrices. Traffic matrices provide information about the flow of packets through routers over time, and we would like to detect outlier traffic (which may represent Denial-of-Service attacks). While nominally similar to the study of time evolution of graphs, there are several special constraints in this dataset.

The primary constraint is speed and scalability: a router under heavy load cannot spend much time to process each packet it handles. Any mining algorithm must operate very quickly on each packet. However, the processing power of a router is also constrained: thus, the computations per packet must be quite basic. We can operate on only a sample of the data, but the realism of this sample also depends on the semantics of the domain: for example, if we want to count the number of open connections, the sample might need to be biased towards `TCP SYN` packets. All of these issues create an extremely interesting problem, and this has attracted a lot of recent interest due to its importance in industry.

Clustering on weighted graphs. Our Cross-associations algorithm operates on unweighted graphs (or equivalently, binary matrices). How can this be extended to weighted graphs?

The main issue is that defining a "homogeneous block" is hard in the weighted context. Again, perhaps we should not be looking for homogeneous blocks at all, but for a "realistic" block such as one generated by, say, R-MAT. This changes the cost function that we optimize using the MDL criterion, and the heuristics for finding a good clustering might also need modifications.

Bibliography

Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 20th VLDB*, pages 487–499, 1994. 5.1

William Aiello, Fan Chung, and Linyuan Lu. A random graph model for massive graphs. In *STOC*, pages 171–180, 2000. 6.1.2, 6.1.2, 6.1.2

William Aiello, Fan Chung, and Linyuan Lu. Random evolution in massive graphs. In *FOCS*, 2001. 6.1.2, 6.1.2, 6.1.2

R. Albert and A.-L. Barabási. Topology of complex networks: local events and universality. *Physical Review Letters*, 85(24), 2000. 6.1.2, 6.1.2, 6.1.2, 6.3

R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–381, 2000. 27, 6.1.2

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 2002. 6.1.1, 6.1.2

N. Alon. Spectral techniques in graph algorithms. In C. L. Lucchesi and A. V. Moura, editors, *Lecture Notes in Computer Science 1380*, pages 206–215. Springer-Verlag, Berlin, 1998. 6.1.1

L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of behavior of small-world networks. In *Proc. Natl. Acad. Sci. U.S.A.*, volume 97, 2000. 6.1.1

Y.W̃eiss Andrew Y.Ñg, Michael I.J̃ordan. On spectral clustering: Analysis and an algorithm. In *Proc. NIPS*, pages 849–856, 2001. 5.1

Chalee Asavathiratham. *The Influence Model: A tractable representation for the dynamics of networked Markov Chains*. PhD thesis, MIT, 2000. 3.1.2

N. T. J. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Hafner Press, 2nd edition, 1975. 3.1.1

A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, pages 509–512, 1999. 3.1.2, 3.4, 6.1.1, 6.1.2, 6.1.2, 6.1.2

A. L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 311, 2002. 6.1.2, 6.1.2, 6.1.2

Albert-László Barabási. *Linked: The New Science of Networks*. Perseus Publishing, first edition, May 2002. 1, 6.1

Asa Ben-Hur and Isabelle Guyon. Detecting stable clusters using principal component analysis. In *Methods in Molecular Biology*, pages 159–182, 2003. 5.1

N. Berger, C. Borgs, J. Chayes, R. M. D'Souza, and R. D. Kleinberg. Degree distribution of competition-induced preferential attachment graphs. *Combinatorics, Probability and Computing*, 2005. 6.1.2

Zhiqiang Bi, Christos Faloutsos, and Flip Korn. The DGX distribution for mining massive, skewed data. In *KDD*, pages 17–26, 2001. 6, 6.1.1, 6.2

G. Bianconi and A.-L. Barabási. Competition and multiscaling in evolving networks. *Europhysics Letters*, 54, 2001. 6.1.2, 6.1.2, 6.1.2

K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, , and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999. 4.1.2

M. Boguñá and R. Pastor-Satorras. Epidemic spreading in correlated complex networks. *Physical Review E*, 66, 2002. 3.1.2

B. Bollobás, C. Borgs, J. Chayes, and O. Riordan. Directed scale-free graphs. In *SIAM Symposium on Discrete Algorithms*, 2003. 6.1.2

P. Bonacich. Power and centrality: a family of measures. *American Journal of Sociology*, 92, 1987. 26

C. Borgs, J. Chayes, M. Mahdian, and A. Saberi. Exploring the community structure of newsgroups (Extended Abstract). In *KDD*, 2004. 6.1.1

S. Boyd, A. Ghosh, B. Prabhakar, and D.Shah. Gossip and mixing times of random walks on random graphs. `www.stanford.edu/~boyd/reports/gossip_opt.pdf`. 4.1.2

U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *European Symposium on Algorithms*, 2003. 6.1.1

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998. 6.1.1

Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web: experiments and models. In *WWW Conf.*, 2000. 6.1.1

T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *INFOCOM*, 2002. 6.1.2, 6.1.2, 6.1.2, 6.3

M. Burns, A. George, and B. Wallace. Simulative performance analysis of gossip failure detection for scalable distributed systems. *Cluster Computing*, 2(2):207–217, 1999. 4.1.2

J. M. Carlson and J. Doyle. Highly optimized tolerance: A mechanism for power laws in designed systems. *Physics Review E*, 60(2):1412–1427, 1999. 6.1.2

Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. R-MAT: A recursive model for graph mining. In *SIAM Data Mining*, 2004. 6.1.1, 6.1.2, 6.1.2, 6.2

H. Chen, J. Schroeder, R. Hauck, L. Ridgeway, H. Atabaksh, H. Gupta, C. Boarman, K. Rasmussen, and A. Clements. COPLINK Connect: Information and knowledge management for law enforcement. *CACM*, 46(1):28–34, January 2003. 1

A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure of very large networks. *Physics Review E*, 70, 2004. 5.1

J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *ICDE*, 2004. 4, 4.1.2

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JASI*, 41(6):391–407, 1990. 5.1

L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1), 1999. 1

A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinchart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC*, 1987. 4.1.2

I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *KDD*, 2003. 5.1, 5.5

Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1–2):143–175, 2001. 5.1

P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001. 3.1.2, 5.5

J. Doyle and J. M. Carlson. Power laws, Highly Optimized Tolerance, and Generalized Source Coding. *Physical Review Letters*, 84(24), June 2000. 6.1.2

Richard Durrett and Xiu-Fang Liu. The contact process on a finite set. *The Annals of Probability*, 16(3):1158–1173, 1988. 3.1.2

Richard Durrett and Roberto H. Schonmann. The contact process on a finite set II. *The Annals of Probability*, 16(4):1570–1583, 1988. 3.1.2

Richard Durrett, Roberto H. Schonmann, and Nelson I. Tanaka. The contact process on a finite set III: The critical case. *The Annals of Probability*, 17(4):1303–1321, 1989. 3.1.2

P. Erdős and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Acadamy of Science*, 5:17–61, 1960. 2.3, 6.1.2, 6.1.2, 6.1.2

Alex Fabrikant, Elias Koutsoupias, and Christos H. Papadimitriou. Heuristically Optimized Trade-offs: A new paradigm for power laws in the Internet (extended abstract), 2002. 6.1.2, 6.1.2, 6.1.2

Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. In *SIGCOMM*, pages 251–262, 1999. 3.1.2, 6.1.1, 6.1.1

Gary William Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of Web communities. In *KDD*, 2000. 5.1, 6.1.1

Nir Friedman, Ori Mosenzon, Noam Slonim, and Naftali Tishby. Multivariate information bottleneck. In *Proc. 17th UAI*, pages 152–161, 2001. 5.1

D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. An empirical study of epidemic algorithms in large scale multihop wireless networks. Technical Report IRB-TR-02-003, Intel Research, 2002. 4.1.2

A. Ganesh, L. Massoulié, and D. Towsley. The effect of network topology on the spread of epidemics. In *INFOCOM*, 2005. 3.3, 4.3

Minos Garofalakis and Amit Kumar. Deterministic wavelet thresholding for maximum-error metrics. In *PODS*, 2004. 4, 4.1.2

P. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. IrisNet: An architecture for a world-wide sensor web. *IEEE Pervasive Computing*, 2003. 4

M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proc. Natl. Acad. Sci. USA*, volume 99, 2002. 5.1

Google Bomb. http://en.wikipedia.org/wiki/Googlebomb. 1

Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for Internet map discovery. In *IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, March 2000. 6.1.1

R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *WWW*, 2004. 4.1.2

Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: an efficient clustering algorithm for large databases. In *Proc. SIGMOD*, pages 73–84, 1998. 5.1

Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, and Robbert van Renesse. Kelips: Building an efficient and stable p2p dht through increased memory and background overhead. In *Proceedings of the International Workshop on Peer-to-Peer Systems*, 2003. 4.1.2

Indranil Gupta, Anne-Marie Kermarrec, and Ayalvadi J. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *SRDS*, 2002. 4.1.2

Gregory Hamerly and Charles Elkan. Learning the k in k-means. In *Proc. 17th NIPS*, 2003. 5.1

Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000. 5.1

Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004. 5.1

T. E. Harris. Contact interactions on a lattice. *Annals of Probability*, 2:969–988, 1974. 3.1.2

Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. 4th KDD*, pages 58–65, 1998. 5.1

M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, 1974. 12, 1, 15, 4

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proc. 22nd SIGIR*, pages 50–57, 1999. 5.1

R. Holman, J. Stanely, and T. Ozkan-Haller. Applying video sensor networks to nearshore environment monitoring. *IEEE Pervasive Computing*, 2(4), 2003. 4

Bret Hull, Kyle Jamieson, and Hari Balakrishnan. Mitigating congestion in wireless sensor networks. In *SenSys*, pages 134–147, 2004. 4.4

Intel. http://db.lcs.mit.edu/labdata/labdata.html. 4.4

G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. Technical Report 98-019, University of Minnesota, 1998a. 6.1.1

George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999. 5.1

George Karypis and Vipin Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Proc. SC98*, pages 1–13, 1998b. 5.1

David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Symposium on Foundations of Computer Science (FOCS)*, 2003. 4.1.2

David Kempe and John Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2002. 4.1.2

David Kempe, John Kleinberg, and Alan Demers. Spatial gossip and resource location protocols. In *ACM Symposium on the Theory of Computing (STOC)*, 2001. 4.1.2

J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *IEEE Symposium on Research in Security and Privacy*, 1991. 3.1.2, 3.5

J. O. Kephart and S. R. White. Measuring and modeling computer virus prevalence. In *IEEE Symposium on Research in Security and Privacy*, 1993. 3.1.2

J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models and methods. In *Proceedings of the International Conference on Combinatorics and Computing*, 1999. 6.1.1, 6.1.2, 6.1.2, 6.1.2

J. M. Kleinberg. Small world phenomena and the dynamics of information. In *NIPS*, 2001. 6.2

Tamara G. Kolda and Dianne P. O'Leary. A semidiscrete matrix decomposition for latent semantic indexing information retrieval. *ACM Transactions on Information Systems*, 16(4):322–346, 1998. 5.1

V. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3), 2001. 1.1

S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *VLDB*, Edinburgh, Scotland, 1999. 6.1.1, 6.1.2, 6.1.2, 6.1.2

Phil Levis, Neil Patel, Scott Shenker, and David Culler. Trickle: A self-regulating mechanism for code propagation and maintenance in wireless networks. In *Proceedings of NSDI*, 2004. 4.1.2

T. M. Liggett. *Interacting Particle Systems*. Springer-Verlag, 1st edition, 1985. 3.1.2

J. Luo, P. Th. Eugster, and J.-P. Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *IEEE INFOCOM*, 2003. 4.1.2

C. R. MacCluer. The many proofs and applications of perron's theorem. *SIAM Review*, 42(3): 487–498, 2000. 3.5

S. Madden, M. Franklin, J. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD*, 2003. 4

N. D. Martinez. Artifacts or Attributes? Effects of resolution on the Little Rock Lake food web. *Ecological Monographs*, 61:367–392, 1991. 1.1

A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. In *SIGCOMM*, pages 18–34, 2000. 6.1.2, 6.1.2, 6.1.2

Milena Mihail and Christos Papadimitriou. On the eigenvalue power law. In *RANDOM*, Harvard, MA, 2002. 3.5

R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovshii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002. 1

Nina Mishra, Dana Ron, and Ram Swaminathan. On finding large conjunctive clusters. In *Proc. 16th COLT*, pages 448–462, 2003. 5.1

A. L. Montgomery and C. Faloutsos. Identifying Web browsing trends and patterns. *IEEE Computer*, 34(7), 2001. 5.5, 6.3

J. Moody. Race, school integration, and friendship segregation in America. *American Journal of Sociology*, 2001. 1.1, 6.1.1, 6.1.1

S. Nath, P. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. In *SenSys*, 2004. 4

M. E. J. Newman, M. Girvan, and J. D. Farmer. Optimal design, robustness and risk aversion. *Physical Review Letters*, 89(2), 2002. 6.1.2

M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64, 2001. 6.1.2, 6.1.2, 6.1.2

C. R. Palmer, P. B. Gibbons, and C. Faloutsos. ANF: A fast and scalable tool for data mining in massive graphs. In *SIGKDD*, Edmonton, AB, Canada, 2002. 6.1.1, 27, 6.1.2

Christopher R. Palmer and J. Gregory Steffan. Generating network topologies that obey power laws. In *GLOBECOM*, November 2000. 6.1.2, 6.1.2, 6.1.2

G. Pandurangan, P. Raghavan, and E. Upfal. Using PageRank to characterize Web structure. In *International Computing and Combinatorics Conf.*, 2002. 6.1.1, 6.1.2, 6.1.2, 6.1.2

Christos H. Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proc. 17th PODS*, pages 159–168, 1998. 5.1

R. Pastor-Satorras, A. Vásquez, and A. Vespignani. Dynamical and correlation properties of the Internet. *Physical Review Letters*, 87(25), 2001. 3.1.2

R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical Review Lettes*, 86(14), 2001. 3.1.2, 3.5

R. Pastor-Satorras and A. Vespignani. Epidemic dynamics in finite size scale-free networks. *Phys. Rev. E*, 65, 2002a. 3.1.2, 3.4

R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. *Phys. Rev. E*, 65, 2002b. 1, 3.1.2

Dan Pelleg and Andrew Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proc. 17th ICML*, pages 727–734, 2000. 5.1

David M. Pennock, Gary W. Flake, Steve Lawrence, Eric J. Glover, and C. Lee Giles. Winners don't take all: Characterizing the competition for links on the Web. *Proceedings of the National Academy of Sciences*, 99(8):5207–5211, 2002. 6.1.1, 6.1.2, 6.1.2, 6.1.2, 6.2, 6.3

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992. 2.4, 2.5

D. S. De S. Price. A general theory of bibliometric and other cumulative advantage processes. *J. Amer. Soc. Inform. Sci.*, 27, 1976. 6.1.2

P. Krishna Reddy and Masaru Kitsuregawa. An approach to relate the web communities through bipartite graphs. In *Proc. 2nd WISE*, pages 302–310, 2001. 5.1

Sidney Redner. How popular is your paper? an empirical study of the citation distribution. *European Physical Journal B*, 4:131–134, 1998. 6.1.1

R. Van Renesse. Scalable and secure resource location. In *Hawaii International Conference on System Sciences*, 2000. 4.1.2

R. Van Renesse, R. Minsky, and M. Hayden. A gossip-style failure detection service. In *Conference on Distributed Systems Platforms and Open Distributed Processing Middleware*, 1998. 4.1.2

Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. In *USENIX*, 2004. 4.1.2

M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70, Edmonton, Canada, 2002. 1, 6.3

M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6 (1), 2002. 3.1.2, 4.4

J. Rissanen. Universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):416–431, 1983. 1

M. F. Schwartz and D. C. M. Wood. Discovering shared interests using graph analysis. *CACM*, 40 (3), 1993. 6.1.1, 6.1.1

H. Simon. On a class of skew distribution functions. *Biometrika*, 42, 1955. 6.1.2

K. Sistla, A. George, R. Todd, and R. Tilak. Performance analysis of flat and layered gossip services for failure detection and consensus in heterogeneous cluster computing. In *Proceedings of the Heterogeneous Computing Workshop (HCW) at the International Parallel and Distributed Processing Symposium (IPDPS)*, 2001. 4.1.2

Chun Tang and Aidong Zhang. Mining multiple phenotype structures underlying gene expression profiles. In *Proc. CIKM03*, pages 418–425, 2003. 5.1

H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topologies, power laws, and hierarchy. Technical Report 01-746, University of Southern California, 2001. 27

S. L. Tauro, C. Palmer, G. Siganos, and M. Faloutsos. A simple conceptual model for the Internet topology. In *Global Internet, San Antonio, Texas*, 2001. 20

Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a dataset via the Gap statistic. *Journal of the Royal Statistical Society, B*, 63:411–423, 2001. 5.1

Alexander Tuzhilin and Gediminas Adomavicius. Handling very large numbers of association rules in the analysis of microarray data. In *Proc. 8th KDD*, pages 396–404, 2002. 5.1

S. M. van Dongen. *Graph clustering by flow simulation*. PhD thesis, Univesity of Utrecht, 2000. 5.1

S.-C. Wang and S.-Y. Kuo. Communication strategies for heartbeat-style failure detectors in wireless ad hoc networks. In *DSN*, 2003. 4.1.2

Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. In *SRDS*, 2003. 3, 4.3

Yang Wang and Chenxi Wang. Modeling the effects of timing parameters on virus propagation. In *WORM*, 2003. 3.1.1

D. J. Watts. *Six Degrees: The Science of a Connected Age*. W. W. Norton and Company, 1st edition, 2003. 6.1

D. J. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296, 2002. 6.2

Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton Univ. Press, 1999. 5.5

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998. 6.1.1, 6.1.2, 6.1.2, 6.1.2

B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9), December 1988. 6.1.2, 6.1.2, 6.1.2

M. R. Weeks, S. Clair, S. P. Borgatti, K. Radda, and J. J. Schensul. Social networks of drug users in high-risk sites: Finding the connections. *AIDS and Behavior*, 6(2), 2002. 1.1

J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, University of Michigan, Ann Arbor, 2002. URL `http://topology.eecs.umich.edu/inet/`. 6.1.2, 6.1.2, 6.1.2

S.-H. Yook, H. Jeong, and A.-L. Barabási. Modeling the Internet's large-scale topology. *Proceedings of the National Academy of Sciences*, 99(21), 2002. 6.1.2, 6.1.2

Bin Zhang, Meichun Hsu, and Umeshwar Dayal. K-harmonic means - a spatial clustering algorithm with boosting. In *Proc. 1st TSDM*, pages 31–45, 2000. 5.1

Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. SIGMOD*, pages 103–114, 1996. 5.1

Shelley Q. Zhuang, Dennis Geels, Ion Stoica, and Randy H. Katz. On failure detection algorithms in overlay networks. In *IEEE INFOCOM*, 2005. 4.1.2